Electronic Transactions on Numerical Analysis.
Volume 65, pp. 156–206, 2026.

ISSN 1068–9613.
DOI: 10.1553/etna_vol65s156

ETNA
Kent State University and
Johann Radon Institute (RICAM)

MONOLITHIC AND BLOCK OVERLAPPING SCHWARZ PRECONDITIONERS FOR THE INCOMPRESSIBLE NAVIER–STOKES EQUATIONS*

ALEXANDER HEINLEIN†, AXEL KLAWONN‡§, JASCHA KNEPPER‡§, AND LEA SAßMANNSHAUSEN‡§

Abstract. Monolithic preconditioners applied to the linear systems arising during the solution of the discretized incompressible Navier–Stokes equations are typically more robust than preconditioners based on incomplete block factorizations. A lower number of iterations and a reduced sensitivity to parameters such as velocity and viscosity can significantly outweigh the additional cost for their setup. Different monolithic preconditioning techniques are introduced and compared to a selection of block preconditioners. In particular, two-level additive overlapping Schwarz methods (OSM) are used to set up monolithic preconditioners and to approximate the inverses arising in the block preconditioners. GDSW-type (Generalized Dryja–Smith–Widlund) coarse spaces are used for the second level. These highly scalable, parallel preconditioners have been implemented in the solver framework `FROSch` (Fast and Robust Overlapping Schwarz), which is part of the software library `Trilinos`. The new GDSW-type coarse space GDSW* is introduced; combining it with other techniques results in a robust algorithm. The block preconditioners PCD (Pressure Convection-Diffusion), SIMPLE (Semi-Implicit Method for Pressure Linked Equations), and LSC (Least-Squares Commutator) are considered to varying degrees. The OSM for the monolithic as well as the block approach allows for the optimized combination of different coarse spaces for the velocity and pressure components, enabling the use of tailored coarse spaces. The numerical and parallel performance of the different preconditioning methods for finite element discretizations of stationary as well as time-dependent incompressible fluid flow problems is investigated and compared. Their robustness is analyzed for a range of Reynolds and Courant–Friedrichs–Lewy (CFL) numbers with respect to a realistic problem setting.

Key words. Navier–Stokes, monolithic preconditioner, block preconditioner, overlapping Schwarz, domain decomposition methods, GDSW-type coarse spaces, PCD, SIMPLE, LSC, Trilinos, FROSch, Teko

AMS subject classifications. 65N55, 65F08, 65-04, 76-04, 76-10

1. Introduction.

Mixed finite element discretizations of the incompressible Navier–Stokes equations lead to systems of saddle-point structure with the unknowns consisting of velocity and pressure variables. We introduce several monolithic preconditioning techniques and present comparisons with a selection of block preconditioners, where one of the key ingredients is finding a suitable approximation of the Schur complement arising from a block factorization. In particular, two-level additive overlapping Schwarz methods are used to set up monolithic preconditioners and to approximate the inverses that appear in the block preconditioners. For the first time, this work combines different types of coarse spaces for the velocity and pressure components of the two-level overlapping Schwarz methods, which improves the robustness for the monolithic preconditioner. By varying Reynolds and CFL numbers in a well-known benchmark problem as well as in a realistic setting, the robustness of the monolithic and block overlapping Schwarz approaches is studied and compared.

We consider the transient and stationary cases of the Navier–Stokes equations. Newton's method is used for linearization, and the finite element method is used to obtain a discrete saddle-point system. The discretization with respect to the velocity and pressure space is

*Received June 25, 2025. Accepted February 9, 2026. Published online on May 28, 2026. Recommended by Andrew J. Wathen.

†Delft Institute of Applied Mathematics, Delft University of Technology, Mekelweg 4, Delft, 2628 CD, Netherlands (a.heinlein@tudelft.nl), https://searhein.github.io.

‡Corresponding author: A. Klawonn. Department of Mathematics and Computer Science, University of Cologne, Weyertal 86–90, Cologne, 50931, Germany
({axel.klawonn, jascha.knepper, l.sassmannshausen}@uni-koeln.de), https://numerik.uni-koeln.de).

§Center for Data and Simulation Science, University of Cologne, Albertus-Magnus-Platz, Cologne, 50923, Germany (https://cds.uni-koeln.de).

ETNA
Kent State University and
Johann Radon Institute (RICAM)

based on P2-P1, Q2-Q1, Q2-P1-disc., or stabilized P1-P1 or Q1-Q1 elements. We solve the linearized saddle-point system using the Generalized Minimal Residual (GMRES) Krylov subspace method [73]. The resulting linear system is ill-conditioned, which motivates the use of suitable preconditioners.

To compare the preconditioners, we show weak scalability results, vary the choice of coarse spaces for the second level of the Schwarz method, and offer recycling strategies for coarse-problem components (for example, the reuse of symbolic factorizations). Furthermore, we show results for a range of Reynolds and CFL numbers for different finite element discretizations and for structured and unstructured geometries and domain decompositions. Finally, we give strong scalability results for one problem.

Various approaches to build efficient discretizations and solvers for the incompressible Navier–Stokes equations exist; see, for example, [5, 6, 7, 13, 26, 35, 36, 38, 39, 56, 63]. Due to the vast number of different possibilities, we limit ourselves to a selection of discretizations, solution processes, and preconditioners. Compared to monolithic preconditioning approaches, there exists extensive research on block preconditioners; see, for example, [9, 14, 26, 28, 29, 30, 53, 54, 55, 58, 59, 75] for a variety of block-preconditioning approaches in the literature.

Since the assembly of the exact Schur complement is usually too expensive, finding a suitable approximation of the Schur complement for the construction of block preconditioners is an integral task; see Section 3. In this work, we consider different Schur complement approximations. For the application of the resulting block preconditioners, the action of the inverses of block matrices is required. These inverses are further approximated with a two-level additive overlapping Schwarz method [76, 77].

The definition of a suitable approximation is highly problem-dependent. For a Stokes problem, a different preconditioner can be used than for a Navier–Stokes problem; in particular, the nonlinearity—influenced by the Reynolds number—plays a vital role. The preconditioning approach can vary if the problem is stationary or time-dependent, if Newton's method is used for linearization or a Picard iteration, if the flow is compressible or incompressible, Newtonian or non-Newtonian, and also the choice of the time and space discretization matters.

When dealing with a steady-state Stokes problem and a stable discretization, the Schur complement can be replaced with a spectrally equivalent scaled pressure mass matrix; see [28, 59]. Consequently, for low Reynolds numbers, we may expect that the pressure mass matrix achieves sufficiently good results. For higher Reynolds numbers, the pressure mass matrix cannot account for the dominating advective forces. Therefore, for larger advective forces, a different approximation of the Schur complement should be used.

The considered block methods are the Pressure Convection-Diffusion (PCD) [14, 29, 31, 51, 53, 74, 75] and Least-Squares Commutator (LSC) [25, 29, 30, 31, 74] triangular block preconditioners and the SIMPLE (Semi-Implicit Method for Pressure Linked Equations) [14, 26, 67, 69, 74] preconditioner. In this work, we restrict ourselves to the SIMPLE variant SIMPLEC and note that variations such as SIMPLER exist; see, for example, [21, 64, 74]. These methods each pursue a different approach to finding a suitable approximation of the Schur complement. In the `Trilinos` package `Teko`, the desired block preconditioners are predefined and can also be combined with other techniques (e.g., algebraic multigrid) for individual blocks [12, 13].

In a monolithic approach, the preconditioner is built for the entire saddle-point system [40, 41, 50, 57, 58]. Consequently, a Schur complement approximation, as for block preconditioners, is not required. Both preconditioning techniques can be extended to handle related problems, such as fluid-structure interaction.

An area of application is the simulation of realistic hemodynamics. This can entail varying viscosities and high velocities leading to Reynolds numbers up to a few thousand; see

ETNA
Kent State University and
Johann Radon Institute (RICAM)

also [14] for parallel preconditioners in the context of hemodynamics. A preconditioner that is robust to changes in the velocity and Reynolds number—for example due to heartbeats—is advantageous. An additional emphasis is placed on achieving parallel scalability.

The remainder of the paper is organized as follows. In Section 2 we introduce the Navier–Stokes equations along with the boundary conditions used in this work, describe the discretization in space and time, and describe the Newton–Krylov method used to solve the nonlinear problem. In Section 3 we derive the block preconditioners that are used later to obtain numerical results. The inverse of the monolithic system and inverses that arise during the construction of block preconditioners are approximated with additive overlapping Schwarz preconditioners; a detailed description is given in Section 4. Different coarse spaces that can be used for the coarse level of the overlapping Schwarz method are specified in Section 4.1. Specifically, we make use of the coarse spaces GDSW [17, 18], RGDSW [20, 41], and we introduce the new variant GDSW*. A description of the parallel implementation is given in Section 5. It consists of the `FEDDLib` [34], which is used for the assembly and as a simulation tool that interfaces `Trilinos`, and the `Trilinos` packages for block and monolithic preconditioning: `Teko` is used for block preconditioners and `FROSch` contains overlapping Schwarz preconditioners used to approximate the inverse of the monolithic system and the inverses arising in the block preconditioners. Section 6 contains results for different problem settings. We consider a backward-facing step geometry for stationary and transient fluid flow and a realistic artery. In both test cases, we consider Reynolds numbers between 200 and 3 200 and a range of CFL numbers to test the robustness of the solvers. The results indicate that, using a combination of the new coarse space GDSW* for the velocity with RGDSW for the pressure, our monolithic preconditioner is more robust than the considered block preconditioners PCD, LSC, and SIMPLEC. In a realistic setting, varying the hemodynamic properties via the viscosity or Reynolds number, the monolithic approach performs best. Specifically, the total number of linear iterations is lower. Despite a higher cost to set up the monolithic preconditioner, the total computation time is usually lower; particularly for high CFL numbers, the difference is significant.

2. The Navier–Stokes equations. We consider the transient, incompressible Navier–Stokes equations to model a Newtonian fluid given by

$$\begin{aligned} \rho\Big(\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u}\cdot\nabla)\mathbf{u}\Big) - \mu\Delta\mathbf{u} + \nabla p &= \rho\mathbf{f}, \\ \operatorname{div}(\mathbf{u}) &= 0, \end{aligned} \tag{2.1}$$

with velocity $\mathbf{u}$, pressure p, density ρ, dynamic viscosity μ, and volume force $\mathbf{f}$. The kinematic viscosity is given by $\nu = \frac{\mu}{\rho}$. If the set of equations (2.1) reaches a steady state for $t \to \infty$, then we have $\partial_t u = 0$ and obtain the stationary Navier–Stokes equations

$$\begin{aligned} \rho(\mathbf{u}\cdot\nabla)\mathbf{u} - \mu\Delta\mathbf{u} + \nabla p &= \rho\mathbf{f}, \\ \operatorname{div}(\mathbf{u}) &= 0. \end{aligned} \tag{2.2}$$

We denote the fluid domain by $\Omega \subset \mathbb{R}^3$ and its boundary by $\partial\Omega$. We assume that there are subsets $\partial\Omega_{\text{in}}, \partial\Omega_{\text{out}} \subset \partial\Omega$, which denote the inflow and outflow parts of the boundary. We assume that $\partial\Omega_{\text{in}}$ and $\partial\Omega_{\text{out}}$ have a positive surface measure. On $\partial\Omega_{\text{in}}$ a velocity field is prescribed for $\mathbf{u}$; on $\partial\Omega_{\text{out}}$ the Neumann boundary condition

$$\nu\nabla\mathbf{u}\cdot\mathbf{n} - p\mathbf{n} = 0 \qquad \text{on } \partial\Omega_{\text{out}}$$

is prescribed. The other boundaries of the fluid domain are walls, $\partial\Omega_{\text{wall}} \subset \partial\Omega$, which prevent fluid from leaving the computational domain. We use a no-slip condition, such that $\mathbf{u} = 0$ on

ETNA
Kent State University and
Johann Radon Institute (RICAM)

$\partial\Omega_{\text{wall}}$. We have

$$\partial\Omega = \partial\Omega_{\text{in}} \cup \partial\Omega_{\text{out}} \cup \partial\Omega_{\text{wall}}$$

and

$$\partial\Omega_{\text{in}} \cap \partial\Omega_{\text{wall}} = \emptyset, \qquad \partial\Omega_{\text{out}} \cap \partial\Omega_{\text{wall}} = \emptyset, \qquad \partial\Omega_{\text{in}} \cap \partial\Omega_{\text{out}} = \emptyset.$$

2.1. Finite element and time discretization. For the spatial discretization of the incompressible fluid flow problem, we use different pairs of mixed finite elements. We partition the domain with characteristic mesh size h into tetrahedra or hexahedra. Let $\tau_h := \tau_h(\Omega)$ denote the triangulation, $\mathcal{P}_k(T)$ the polynomial space of order k, and $\mathcal{Q}_k(T)$ the polynomial space of order up to k in each coordinate direction. For the partition into tetrahedral elements, we use continuous and either piecewise linear or quadratic functions for the velocity space,

$$\mathbf{V}_{Pk}^h(\Omega) = \Big\{\mathbf{v}_h \in \big(C(\Omega)\big)^3 \cap \big(H^1(\Omega)\big)^3 : \mathbf{v}_h|_T \in \big(\mathcal{P}_k(T)\big)^3 \; \forall T \in \tau_h\Big\}, \quad k \in \{1,2\}.$$

Similarly, we can define the velocity space for hexahedral elements,

$$\mathbf{V}_{Qk}^h(\Omega) = \Big\{\mathbf{v}_h \in \big(C(\Omega)\big)^3 \cap \big(H^1(\Omega)\big)^3 : \mathbf{v}_h|_T \in \big(\mathcal{Q}_k(T)\big)^3 \; \forall T \in \tau_h\Big\}, \quad k \in \{1,2\}.$$

The continuous and piecewise linear pressure spaces for the different partitions are defined as

$$Q_{P1}^h(\Omega) = \Big\{q_h \in C(\Omega) \cap L^2(\Omega) : q_h|_T \in \mathcal{P}_1(T) \; \forall T \in \tau_h\Big\},$$
$$Q_{Q1}^h(\Omega) = \Big\{q_h \in C(\Omega) \cap L^2(\Omega) : q_h|_T \in \mathcal{Q}_1(T) \; \forall T \in \tau_h\Big\}.$$

Additionally, we consider the combination of Q2-P1-disc., where the pressure space consists of discontinuous piecewise linear functions:

$$Q_{\text{P1-disc.}}^h(\Omega) = \Big\{q_h \in L^2(\Omega) : q_h|_T \in \mathcal{P}_1(T) \; \forall T \in \tau_h\Big\}.$$

If the type of element is unspecified, then we write $\mathbf{V}^h$ and Q^h for the velocity and pressure space, respectively. Let $\mathbf{u}_h \in \mathbf{V}^h$ and $p_h \in Q^h$ be the approximations of the solutions $\mathbf{u}$ and p. Then the resulting variational formulation for the stationary case (2.2) is as follows:

Find $\mathbf{u}_h \in \mathbf{V}^h$ and $p_h \in Q^h$ such that

$$\nu \int_\Omega \nabla\mathbf{u}_h : \nabla\mathbf{v}_h \,\mathrm{d}x + \int_\Omega ((\mathbf{u}_h \cdot \nabla)\mathbf{u}_h) \cdot \mathbf{v}_h \,\mathrm{d}x - \int_\Omega p_h(\nabla \cdot \mathbf{v}_h)\,\mathrm{d}x = \int_\Omega \mathbf{f} \cdot \mathbf{v}_h \,\mathrm{d}x \quad \forall \mathbf{v}_h \in \mathbf{V}^h,$$
$$\int_\Omega q_h(\nabla \cdot \mathbf{u}_h)\,\mathrm{d}x = 0 \qquad \forall q_h \in Q^h.$$

We denote by $\boldsymbol{\varphi}_1, \ldots, \boldsymbol{\varphi}_{N_u}$ the finite element basis of $\mathbf{V}^h$ and by $\psi_1, \ldots, \psi_{N_p}$ the basis of Q^h. Then, we can write

$$\mathbf{u}_h(x) = \sum_{i=1}^{N_u} u_i \boldsymbol{\varphi}_i(x) \qquad \text{and} \qquad p_h(x) = \sum_{i=1}^{N_p} p_i \psi_i(x).$$

We will also refer to the corresponding coefficient vectors of $\mathbf{u}_h(x)$ and $p_h(x)$ as $\mathbf{u}_h$ and p_h, respectively. We define the following matrix operators corresponding to the weak formulation,

$$A = [a_{ij}], \qquad B = [b_{ij}], \qquad N(\mathbf{u}_h) = [n_{ij}(\mathbf{u}_h)],$$
$$a_{ij} = \int_\Omega \nabla\boldsymbol{\varphi}_i : \nabla\boldsymbol{\varphi}_j \,\mathrm{d}x, \quad b_{ij} = -\int_\Omega \psi_i(\nabla \cdot \boldsymbol{\varphi}_j)\,\mathrm{d}x, \quad n_{ij}(\mathbf{u}_h) = \int_\Omega ((\mathbf{u}_h \cdot \nabla)\boldsymbol{\varphi}_j) \cdot \boldsymbol{\varphi}_i \,\mathrm{d}x$$

ETNA
Kent State University and
Johann Radon Institute (RICAM)

and obtain with the discretization $\mathbf{f}_u$ of $\mathbf{f}$ the system of linear equations for the stationary Navier–Stokes equations,

$$\begin{bmatrix} F_O(\mathbf{u}_h) & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} \mathbf{u}_h \\ p_h \end{bmatrix} = \begin{bmatrix} \mathbf{f}_u \\ 0 \end{bmatrix}, \tag{2.3}$$

with $F_O(\mathbf{u}_h) = \nu A + N(\mathbf{u}_h)$. The resulting Picard iteration with the iterates $\mathbf{u}_h^k$ and p_h^k reads [29, p. 346]

$$\begin{bmatrix} F_O(\mathbf{u}_h^k) & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} \mathbf{u}_h^{k+1} \\ p_h^{k+1} \end{bmatrix} = \begin{bmatrix} \mathbf{f}_u \\ 0 \end{bmatrix},$$

defining the discretized Oseen problem. We apply the Newton method to solve the nonlinear problem:

$$\begin{aligned} &\text{Solve:} && \mathcal{J}(X^k)\delta X^k = -\mathcal{R}(X^k), \\ &\text{Update:} && X^{k+1} = X^k + \delta X^k, \end{aligned}$$

where the δ-notation is defined as $\delta X^k := X^{k+1} - X^k$. The Newton residual $\mathcal{R}$ for the stationary case is defined as

$$\mathcal{R}_{\text{stat}}\left(\mathbf{u}_h^k, p_h^k\right) = \begin{bmatrix} F_O(\mathbf{u}_h^k) & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} \mathbf{u}_h^k \\ p_h^k \end{bmatrix} - \begin{bmatrix} \mathbf{f}_u \\ 0 \end{bmatrix}.$$

We need to linearize (2.3) by adding an additional term $W(\mathbf{u}_h^k)$ to $F_O(\mathbf{u}_h^k)$ from the derivative of the convection term,

$$W(\mathbf{u}_h^k) = [w_{ij}(\mathbf{u}_h^k)], \qquad w_{ij}(\mathbf{u}_h^k) = \int_\Omega ((\boldsymbol{\varphi}_j \cdot \nabla)\mathbf{u}_h^k) \cdot \boldsymbol{\varphi}_i \, \mathrm{d}x,$$

with the velocity solution from the previous kth Newton step $\mathbf{u}_h^k$. This leads to

$$\mathcal{J}\left(\mathbf{u}_h^k\right) \begin{bmatrix} \delta\mathbf{u}_h^k \\ \delta p_h^k \end{bmatrix} = \begin{bmatrix} F_N(\mathbf{u}_h^k) & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} \delta\mathbf{u}_h^k \\ \delta p_h^k \end{bmatrix} = -\mathcal{R}_{\text{stat}}\left(\mathbf{u}_h^k, p_h^k\right),$$

with $F_N(\mathbf{u}_h^k) = F_O(\mathbf{u}_h^k) + W(\mathbf{u}_h^k)$. For transient problems we include the BDF-2 time discretization [37] and expand the system to

$$\mathcal{J}\left(\mathbf{u}_h^{k,n+1}\right) \begin{bmatrix} \delta\mathbf{u}_h^{k,n+1} \\ \delta p_h^{k,n+1} \end{bmatrix} = -\mathcal{R}_{\text{stat}}\left(\mathbf{u}_h^{k,n+1}, p_h^{k,n+1}\right) + \begin{bmatrix} \frac{2}{\Delta t} M_u \mathbf{u}_h^n - \frac{1}{2\Delta t} M_u \mathbf{u}_h^{n-1} \\ 0 \end{bmatrix}, \tag{2.4}$$

where M_u is the velocity mass matrix, $\mathbf{u}_h^n$ and $\mathbf{u}_h^{n-1}$ are the solutions of the last two time steps, and

$$\mathcal{J}\left(\mathbf{u}_h^{k,n+1}\right) = \begin{bmatrix} \frac{3}{2\Delta t_n} M_u + F_N(\mathbf{u}_h^{k,n+1}) & B^T \\ B & 0 \end{bmatrix}.$$

The resulting saddle-point problem for the different discretizations has the generic form

$$\mathcal{F} := \begin{bmatrix} F & B^T \\ B & -C \end{bmatrix}, \tag{2.5}$$

ETNA
Kent State University and
Johann Radon Institute (RICAM)

where the fluid component F depends on the underlying problem setting and $C \neq 0$ in case of a discretization that is not inf-sup stable.

We focus on three-dimensional problems for which the P2-P1, Q2-Q1, and Q2-P1-disc. finite element pairs satisfy the inf-sup condition; cf. [8]. In case of the non inf-sup stable finite elements P1-P1 and Q1-Q1, the Bochev–Dohrmann stabilization [16] is used, which penalizes nonphysical pressure variations by inserting a pressure term into the conservation of mass equation (see [16, Section 2] and [50, Section 5.6.1]). Its bilinear form reads

$$C(p,q) = \frac{1}{\nu}\int_\Omega (p-\rho_0 p)(q-\rho_0 q)\,\mathrm{d}x \qquad \forall q \in L^2(\Omega),$$

where ρ_0 is an L^2-orthogonal projection onto the space $\mathcal{P}_0(\tau_h)$ of discontinuous, piecewise constant functions; the projection is given by

$$(\rho_0 q)|_T = \left(\int_T 1\,\mathrm{d}x\right)^{-1}\int_T q\,\mathrm{d}x \qquad \forall T \in \tau_h(\Omega),$$

which can be computed efficiently due to its independent definition on each element $T \in \tau_h(\Omega)$.

As a measure that indicates how much information travels through a computational grid cell per unit time, that is, a relation of the velocity of transport, time step, and mesh resolution, we define a Courant(–Friedrichs–Lewy)-type number (CFL) for structured meshes as

$$c_{\mathrm{CFL}}(t) = \frac{u_\infty^h(t)\Delta t}{h}, \qquad u_\infty^h(t) = \max_{x^h \in \tau_h(\Omega)} \|\mathbf{u}_h(x^h,t)\|_2,$$

where x^h is a finite element node. For an unstructured mesh, we take varying element sizes and shapes into account and use

$$c_{\mathrm{CFL}}(t) = \max_{T\in\tau_h(\Omega)} c_{\mathrm{CFL},T}(t),$$
$$c_{\mathrm{CFL},T}(t) = \frac{u_\infty^h(T,t)\Delta t}{d_I(T)} \qquad \text{with} \qquad u_\infty^h(T,t) = \max_{x^h\in T}\|\mathbf{u}_h(x^h,t)\|_2,$$

where $T \in \tau_h(\Omega)$ is a finite element and $d_I(T)$ is the diameter of the largest incircle of T.

We will also use the average of $c_{\mathrm{CFL},T}(t)$ over all elements $T \in \tau_h(\Omega)$, denoted by $c_{\mathrm{CFL,avg}}(t)$. By c_{CFL} and $c_{\mathrm{CFL,avg}}$, we refer to the corresponding maximum CFL numbers over time. In case the CFL number is too large, instability or unphysical behavior may be encountered. Generally, implicit time-stepping schemes are more stable with respect to the CFL number. High CFL numbers tend to be more demanding for preconditioners; see, for example, [12, Figure 9] and Figures 6.13 and 6.14.

2.2. The inexact Newton method: Newton–Krylov. We apply an inexact Newton method [23] by solving the linearized system with a Krylov subspace method up to a certain tolerance; specifically, we employ the generalized minimal residual (GMRES) method [73]. With an initial guess for δX^k, we use GMRES to solve

$$\|\,\mathcal{J}(X^k)\delta X^k + \mathcal{R}(X^k)\|_2 \leq \eta_k \|\,\mathcal{R}(X^k)\|_2, \qquad \eta_k \in [0,1),$$

and update the Newton iterate via $X^{k+1} = X^k + \delta X^k$. The linear solver tolerance η_k can be constant throughout the Newton iterations or change between consecutive iterations. In the second case, η_k is referred to as forcing term [24], which can change depending on the underlying strategy. We use the *Choice 2* forcing term from [70],

$$\eta_k = \gamma\left(\frac{\|\,\mathcal{R}(X^k)\|_2}{\|\,\mathcal{R}(X^{k-1})\|_2}\right)^\alpha,$$

ETNA
Kent State University and
Johann Radon Institute (RICAM)

with $\gamma \in [0,1]$ and $\alpha \in (1,2]$. To make use of this strategy, we select an $\eta_0 \in [0,1)$ for the first iteration. We impose a lower and upper bound for the forcing term by

$$\max\left\{\gamma\eta_{k-1}^{\alpha}, \eta_{\min}\right\} \leq \eta_k \leq \eta_{\max}, \tag{2.6}$$

with prescribed bounds $\eta_{\min}$ and $\eta_{\max}$.

3. Block preconditioners. We introduce a block LDU decomposition of the Navier–Stokes saddle-point problem (2.5),

$$\mathcal{F} = \begin{bmatrix} F & B^T \\ B & -C \end{bmatrix} = \begin{bmatrix} I & 0 \\ BF^{-1} & I \end{bmatrix} \begin{bmatrix} F & 0 \\ 0 & S \end{bmatrix} \begin{bmatrix} I & F^{-1}B^T \\ 0 & I \end{bmatrix}, \tag{3.1}$$

where $S = -C - BF^{-1}B^T$ is the Schur complement. Generally, we seek to replace F and the Schur complement S by suitable approximations $\hat{F}$ and $\hat{S}$ that facilitate the computation of $\hat{F}^{-1}$ and $\hat{S}^{-1}$. Suitable approximations depend on the problem setting. The approximation of F^{-1} will be based on a Schwarz preconditioner (see Section 4) and the approximation of S^{-1} on the approaches LSC, PCD, SIMPLE, and SIMPLEC. These preconditioners contain inverses of matrices that will be further approximated with a Schwarz method. A resulting upper block-triangular preconditioner reads

$$\mathcal{B}_{\text{Tri}}^{-1} = \begin{bmatrix} \hat{F} & B^T \\ 0 & \hat{S} \end{bmatrix}^{-1} = \begin{bmatrix} \hat{F}^{-1} & -\hat{F}^{-1}B^T\hat{S}^{-1} \\ 0 & \hat{S}^{-1} \end{bmatrix}, \tag{3.2}$$

which encapsulates the process of block Gaussian elimination.

For a Navier–Stokes problem with kinematic viscosity ν, velocity v, and a characteristic length L (for example, the diameter of a vessel), the Reynolds number is defined as

$$\text{Re} = \frac{vL}{\nu}. \tag{3.3}$$

For steady-state Stokes problems and a stable discretization, the Schur complement can be replaced with the spectrally equivalent, scaled pressure mass matrix $-\frac{1}{\nu}M_p$ [28, 59]. We may thus expect that this approach also works well for small Reynolds numbers. For higher Reynolds numbers, the pressure mass matrix cannot account for the dominating advective forces in F. Therefore, for larger advective forces, a different approximation of the Schur complement should be used. We present the PCD (Pressure Convection-Diffusion) and LSC (Least-Squares Commutator) block-triangular preconditioners and the SIMPLE method (Semi-Implicit Method for Pressure Linked Equations) in the following sections.

The PCD and LSC Schur complement approximations are motivated by finding a suitable approximation of F^{-1}, which is used to construct the Schur complement. Here and in the following, for the construction of approximations of F^{-1}, we assume that $F = F_O$, the discrete Oseen operator. In an implementation, however, we will use the general F in (2.5) that includes an additional term from the Newton linearization. For the time discretization, see (3.9).

The discrete Oseen operator is the discretization of a convection-diffusion differential operator for the velocity,

$$\mathcal{L} := -\nu\Delta + \mathbf{w}_h \cdot \nabla,$$

where $\mathbf{w}_h$ is the approximation to the velocity solution from the previous nonlinear iteration. The discretization of $\mathcal{L}$ corresponds to $F(\mathbf{w}_h) = \nu A + N(\mathbf{w}_h)$. We denote the corresponding differential operator on the pressure space by

$$\mathcal{L}_p := (-\nu\Delta + \mathbf{w}_h \cdot \nabla)_p.$$

ETNA
Kent State University and
Johann Radon Institute (RICAM)

TABLE 3.1
Operators appearing in the different preconditioning strategies. *These operators are usually further approximated; for example, F^{-1} can be approximated with a Schwarz method. Often the inverse of a mass matrix M is replaced by $H_M^D = \operatorname{diag}(M)^{-1}$, the inverse of the diagonal entries of M. Then, an elaborate strategy (e.g., a Schwarz method) for the approximation of M^{-1} is not necessary.*

Operator	Monolithic	PCD	SIMPLE/SIMPLEC	LSC	LSC$_{A_p}$	LSC$_{\text{stab},A_p}$
$\mathcal{F}^{-1}$	✓					
F^{-1}		✓	✓	✓	✓	✓
A_p^{-1}		✓			✓	✓
M_p^{-1}		✓				
M_u^{-1}				✓	✓	✓
$\rho_{\text{eig}}^{-1} D^{-1}$						✓
$(BM_u^{-1}B^T)^{-1}$				✓		
$(-C - BH_F B^T)^{-1}$			✓			

Its discretization is given by $F_p(\mathbf{w}_h) = \nu A_p + N_p(\mathbf{w}_h) = [f_{p,ij}(\mathbf{w}_h)]$, with

$$f_{p,ij}(\mathbf{w}_h) = \nu \int_\Omega \nabla\psi_j \cdot \nabla\psi_i \,\mathrm{d}x + \int_\Omega (\mathbf{w}_h \cdot \nabla\psi_j)\psi_i \,\mathrm{d}x.$$

Using the convection-diffusion operators, we can define a commutator [29, (9.13)]

$$\mathcal{E} := \nabla \cdot (-\nu\Delta + \mathbf{w}_h \cdot \nabla) - (-\nu\Delta + \mathbf{w}_h \cdot \nabla)_p \nabla \cdot \tag{3.4}$$

to find an approximation $\hat{S}$ of the Schur complement S. Note that it is assumed that the differential operators are applied to functions that are sufficiently regular; they are not well defined for the finite element spaces used.

We suppose that the commutator (3.4) is small in some sense in order to define the PCD and LSC preconditioners. The PCD Schur complement approximation is derived from a discretization of the commutator. In turn, to define LSC, the matrix F_p is replaced with a new matrix that makes (3.4) small in a least-squares sense.

Table 3.1 lists the matrices that are required to set up the different preconditioners that are introduced in the following sections; these matrices are usually further approximated via, for example, lumping or a Schwarz method.

3.1. PCD (Pressure Convection-Diffusion). We present a derivation following [74] and [29, Chapter 9] of the pressure convection-diffusion preconditioner, which was introduced in [53, 75].

In the velocity space with zero boundary conditions, the matrix representation of the discrete divergence operator is $-M_p^{-1}B$, and $M_u^{-1}B^T$ is the representation of the discrete gradient operator [29, p. 409]. M_u is the velocity mass matrix and M_p the pressure mass matrix:

$$M_u = [m_{ij}^u], \qquad m_{ij}^u = \int_\Omega \boldsymbol{\varphi}_i \cdot \boldsymbol{\varphi}_j \,\mathrm{d}x, \qquad M_p = [m_{ij}^p], \qquad m_{ij}^p = \int_\Omega \psi_i\psi_j \,\mathrm{d}x.$$

With these representations, we can define a discrete version of the commutator (3.4):

$$\mathcal{E}_h = (M_p^{-1}B)(M_u^{-1}F) - (M_p^{-1}F_p)(M_p^{-1}B). \tag{3.5}$$

ETNA
Kent State University and
Johann Radon Institute (RICAM)

Suppose that the commutators $\mathcal{E}$ and $\mathcal{E}_h$ are small in some operator norm (cf. [29, Remark 9.5] and Section 3.2). Pre-multiplication of (3.5) with $M_p F_p^{-1} M_p$, post-multiplication with $F^{-1}B^T$, and assuming that the commutator is small leads to the Schur complement approximation

$$-BF^{-1}B^T \approx -M_p F_p^{-1} B M_u^{-1} B^T. \tag{3.6}$$

Note that F is not symmetric and that, in this work, the approximation sign is not meant to denote spectral equivalence but that the difference is approximately zero in some matrix norm. We require the application of the inverse of the Schur complement, that is, also of its approximation $BM_u^{-1}B^T$. This expensive part is replaced with the pressure-Laplace operator A_p, which is spectrally equivalent for an enclosed-flow problem (cf. [29, p. 366] and [74]):

$$\begin{aligned} q_h^T A_p p_h = \langle \nabla p_h, \nabla q_h \rangle_{L_2} &\approx \langle M_u^{-1} B^T p_h, M_u^{-1} B^T q_h \rangle_{M_u} \\ &= \langle B^T p_h, M_u^{-1} B^T q_h \rangle_{l_2} \\ &= q_h^T B M_u^{-1} B^T p_h. \end{aligned}$$

This gives us

$$S = -BF^{-1}B^T \approx -M_p F_p^{-1} A_p =: S_{\text{PCD}}. \tag{3.7}$$

Then, we can define the PCD block-triangular preconditioner

$$\mathcal{B}_{\text{PCD}}^{-1} = \begin{bmatrix} F & B^T \\ 0 & S_{\text{PCD}} \end{bmatrix}^{-1}.$$

As in [75] we can extend the operator F_p to also reflect the contribution of the BDF-2 time discretization for a transient Navier–Stokes problem:

$$F_p(\mathbf{w}_h) = \frac{3}{2\Delta t} M_p + N_p(\mathbf{w}_h) + \nu A_p. \tag{3.8}$$

The motivation is similar to before; the commutator is extended to

$$\mathcal{E} = \nabla \cdot \left(\frac{\partial}{\partial t} - \nu\Delta + \mathbf{w}_h \cdot \nabla \right) - \left(\frac{\partial}{\partial t} - \nu\Delta + \mathbf{w}_h \cdot \nabla \right)_p \nabla \cdot \tag{3.9}$$

to include a time derivative.

Note that the definition of S_{PCD} is based on the assumption of a small error between the operators $\mathcal{L}$ and $\mathcal{L}_p$. This is only directly applicable to the fixed-point method as Newton's method introduces an additional term for the linearization. Consequently for Newton's method, we cannot necessarily assume the error as defined above to be small. Nonetheless, it is observed that Newton's method combined with PCD preconditioning yields good results; see, for example, [29, Chapters 9.3, 9.4].

3.1.1. Stabilized discretizations. For elements that are not inf-sup stable, the PCD preconditioner also extends to stabilized approximations (e.g., P1-P1); see [29, Chapter 9]. A small commutator $\mathcal{E}$ (see (3.4)) implies

$$\nabla \cdot (-\nu\Delta + \mathbf{w}_h \cdot \nabla)^{-1} \nabla \approx (-\nu\Delta + \mathbf{w}_h \cdot \nabla)_p^{-1} \nabla \cdot \nabla. \tag{3.10}$$

Similarly to before, the left-hand side of (3.10) can be expressed discretely as

$$(-M_p^{-1}B)(M_u^{-1}F)^{-1}(M_u^{-1}B^T) = M_p^{-1}(-BF^{-1}B^T).$$

ETNA
Kent State University and
Johann Radon Institute (RICAM)

This is the scaled Schur complement matrix in the case of unstabilized mixed finite element approximations. If our discretization is stabilized, then the Schur complement expands to $S = -BF^{-1}B^T - C$ so that we obtain

$$M_p^{-1}(-BF^{-1}B^T - C)$$

for the left-hand side of (3.10). For the right-hand side we get

$$(M_p^{-1}F_p)^{-1}(-M_p^{-1}B)(M_u^{-1}B^T) = F_p^{-1}(-BM_u^{-1}B^T).$$

Thus, for stabilized elements, we have

$$M_p^{-1}(-BF^{-1}B^T - C) \approx F_p^{-1}(-BM_u^{-1}B^T).$$

The matrix $-BM_u^{-1}B^T$ stems from the discretization of $\nabla \cdot \nabla$, but it is not a good representation of a discrete Laplacian in cases where the underlying mixed approximation is not uniformly stable [29, Chapter 9, p. 369]. Furthermore, for inherently unstable discretizations, highly oscillating components are in the null space of B^T, and, consequently, $BM_u^{-1}B^T$ is singular. However, we can fix this by using $-A_p$ as a more direct discretization of Δ as described in the previous section. As a result, in the stabilized case, we obtain the Schur complement approximation $S \approx -M_pF_p^{-1}A_p = S_{\text{PCD}}$, which is the same as in the stable case; cf. [12, p. 351].

3.1.2. Boundary conditions. The assembly of the original Schur complement S contains information about the boundary conditions. Similarly, we need to account for this information in the components of S_{PCD}. For a detailed discussion on this topic, we refer to [29, 31], where different boundary conditions are considered. Originally in [53], F_p and A_p were constructed with Neumann, i.e., do-nothing boundary conditions. Then, based on the analysis of a one- and two-dimensional problem, other boundary conditions were derived; see [29, 31].

The derivation of the preconditioner using homogeneous Neumann boundary conditions is based on enclosed-flow problems (e.g., lid-driven cavity). Consequently, we should expect it to perform better in these types of problems compared to non-enclosed-flow problems. In inflow and outflow problems, A_p and F_p can be treated using different boundary conditions to improve performance. Note that A_p is not only used in the PCD Schur complement approximation $-M_pF_p^{-1}A_p$ but also for the assembly of $F_p = \frac{3}{2\Delta t}M_p + N_p + \nu A_p$. The setup of the two instances is based on the same Neumann matrix corresponding to A_p, but, subsequently, boundary conditions are set separately for A_p in S_{PCD} and for F_p. We review a few results from [29, 31, 51]. The main idea is to construct F_p (3.8) and A_p as though they come from the corresponding boundary value problem. For the discrete Laplace operator A_p, forcing the discrete pressure p_h to satisfy a homogeneous Dirichlet boundary condition along the outflow boundary of $\partial\Omega_N$ is recommended; see [29]. For the $\partial\Omega_D = \partial\Omega \setminus \partial\Omega_N$-part of the boundary of the domain, a Neumann condition $\frac{\partial p_h}{\partial \mathbf{n}} = 0$ should be applied in A_p. For the discrete convection-diffusion operator F_p, it is suggested that the Robin condition

$$-\nu\frac{\partial p_h}{\partial \mathbf{n}} + (\mathbf{w}_h \cdot \mathbf{n})p_h = 0$$

for all boundary edges is appropriate; see, for example, [29, 31]. Note that this includes edges where Dirichlet velocity boundary conditions are applied, such as the inflow boundary or the walls. The use of Dirichlet conditions for F_p along the outlet $\partial\Omega_N$ can be beneficial. In [31, Section 7] different boundary conditions for F_p were discussed, and the use of a Robin boundary condition along the inlet and Dirichlet boundary conditions along the outlet showed

ETNA
Kent State University and
Johann Radon Institute (RICAM)

TABLE 3.2
Boundary conditions used in the PCD preconditioner. *Different strategies for setting homogeneous boundary conditions in A_p and F_p denoted as D: Dirichlet, N: Neumann, and R: Robin. The default strategy (BC-3) is highlighted.*

	A_p			F_p		
	$\partial\Omega_{\text{out}}$	$\partial\Omega_{\text{in}}$	$\partial\Omega_{\text{wall}}$	$\partial\Omega_{\text{out}}$	$\partial\Omega_{\text{in}}$	$\partial\Omega_{\text{wall}}$
(BC-1)	D	N	N	D	N	N
(BC-2)	D	N	N	D	R	N
(BC-3)	D	N	N	N	R	N

good results for Reynolds numbers between 10 and 100. For higher Reynolds numbers in the range of 200 to 400, results for Robin on the inlet and Neumann boundary conditions on the outlet proved best. In [13], using a similar implementation to ours, the use of scaled Dirichlet conditions was motivated. Other scaling options are mentioned in [31].

Since it is widely emphasized [13, 29, 31, 51] that the boundary conditions applied to PCD can have a substantial effect on the convergence (see also Table 6.7), we will consider the options in Table 3.2 for simulations but keep the recommended one (BC-3) as a default, where A_p has a Dirichlet condition on the outlet and F_p a Robin condition on the inlet.

3.2. LSC (Least-Squares Commutator). The construction of the LSC preconditioner follows [25, 30] and [29, Section 9.2.3]. Similarly to before, we try to define a suitable approximation of the Schur complement by reducing the error of a commutator. Unlike in Section 3, however, we do not use the corresponding discrete commutator directly to define a Schur complement approximation with the help of a newly assembled matrix F_p. Instead, F_p is replaced by a matrix that does not require the assembly of an additional matrix (we want to avoid having to assemble the discrete convection-diffusion operator for the pressure), and this results in the commutator being small with respect to some norm.

The commutator $\varepsilon = \nabla\cdot(-\nu\Delta + \mathbf{w}_h\cdot\nabla) - (-\nu\Delta + \mathbf{w}_h\cdot\nabla)_p\nabla\cdot$ from Section 3 can be regarded as the adjoint commutator to (cf. [29, Remark 9.3])

$$\varepsilon^* := (-\nu\Delta - \mathbf{w}_h\cdot\nabla)\nabla - \nabla(-\nu\Delta - \mathbf{w}_h\cdot\nabla)_p.$$

This time we base the approximation on the minimization of the commutator in the least-squares sense. Specifically, we seek to minimize the operator norm

$$\sup_{p_h\neq 0}\frac{\|\varepsilon^* p_h\|_{L^2(\Omega)}}{\|p_h\|_{L^2(\Omega)}}.$$

Similar to before, we can formulate this in a discrete sense and minimize

$$\sup_{p_h\neq 0}\frac{\|\varepsilon_h^* p_h\|_{M_u}}{\|p_h\|_{M_p}}, \qquad \varepsilon_h^* = (M_u^{-1}F)(M_u^{-1}B^T) - (M_u^{-1}B^T)(M_p^{-1}F_p), \tag{3.11}$$

where in this context p_h corresponds to its coefficient vector, $\|\mathbf{v}\|_{M_u} = \langle M_u\mathbf{v},\mathbf{v}\rangle^{\frac{1}{2}}$, and $\|p_h\|_{M_p} = \langle M_p p_h, p_h\rangle^{\frac{1}{2}}$. Instead of using F_p as the discrete convection-diffusion operator for the pressure, we replace it with a matrix that yields a small (not necessarily minimal) value for (3.11). One way to achieve this is to minimize the individual vector norms of the columns of the discrete commutator, that is, by defining the jth column $[F_p]_j$ of F_p to solve the weighted least-squares problem

$$\min_{[F_p]_j}\left\|\left[M_u^{-1}FM_u^{-1}B^T\right]_j - M_u^{-1}B^T M_p^{-1}\,[F_p]_j\right\|_{M_u}.$$

ETNA
Kent State University and
Johann Radon Institute (RICAM)

The associated normal equations are

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

which leads to the following definition of F_p:

$$F_p = M_p(BM_u^{-1}B^T)^{-1}(BM_u^{-1}FM_u^{-1}B^T). \tag{3.12}$$

Similarly to how we have derived the Schur complement approximation (3.6), we can pre-multiply ε_h^* with $BF^{-1}M_u$ and post-multiply it with $F_p^{-1}M_p$. Under the assumption that the commutator is small, we obtain

$$-BF^{-1}B^T \approx -BM_u^{-1}B^T F_p^{-1}M_p.$$

Substituting F_p from (3.12) into this Schur complement approximation gives

$$-BF^{-1}B^T \approx -\left(BM_u^{-1}B^T\right)\left(BM_u^{-1}FM_u^{-1}B^T\right)^{-1}\left(BM_u^{-1}B^T\right).$$

The approximate inverse of the Schur complement for the LSC block preconditioner for a stable discretization reads

$$S_{\text{LSC}}^{-1} := -(BM_u^{-1}B^T)^{-1}(BM_u^{-1}FM_u^{-1}B^T)(BM_u^{-1}B^T)^{-1}. \tag{3.13}$$

We define the LSC block-triangular preconditioner

$$\mathcal{B}_{\text{LSC}}^{-1} = \begin{bmatrix} F & B^T \\ 0 & S_{\text{LSC}} \end{bmatrix}^{-1}.$$

If the mass matrix M_u is replaced with the identity matrix, then the resulting Schur complement approximation,

$$S_{\text{BFBt}}^{-1} = -(BB^T)^{-1}(BFB^T)(BB^T)^{-1}, \tag{3.14}$$

is the classical BFBt preconditioner [30], as opposed to the LSC preconditioner being a scaled BFBt method.

As for the construction of the PCD preconditioner, $BM_u^{-1}B^T$ can be replaced with the pressure-Laplace matrix:

$$S_{\text{LSC}_{A_p}}^{-1} = -A_p^{-1}(BM_u^{-1}FM_u^{-1}B^T)A_p^{-1}. \tag{3.15}$$

We have chosen to use this replacement strategy in our simulations; cf. Section 5.3.2. As before, A_p can be assembled using different boundary conditions [29, Section 9.2.2]; see also [31]. In this work, we will only use (BC-2) for the LSC variants, that is, a Dirichlet condition at the outflow of the domain.

Stabilization. In non inf-sup stable discretizations, the LSC approach requires some modification. As mentioned in Section 3.1.1, the approximation $BM_u^{-1}B^T$ of the discrete Laplacian is singular due to the nontrivial null space of B^T. In [27, Section 4.2] (see also [12, Section 3.3.3] and [26, Section 3.2.2]), a modification was proposed by adding scaled stabilization operators,

$$S_{\text{LSC}_{\text{stab}}}^{-1} = -(BM_u^{-1}B^T + \gamma C)^{-1}(BM_u^{-1}FM_u^{-1}B^T)(BM_u^{-1}B^T + \gamma C)^{-1} - \alpha D^{-1},$$

ETNA
Kent State University and
Johann Radon Institute (RICAM)

where, using $\operatorname{diag}(\cdot)$ for the diagonal part and $\rho_{\text{eig}}(\cdot)$ for the spectral radius of a matrix,

$$\gamma := \frac{\rho_{\text{eig}}\left(M_u^{-1}F\right)}{3},$$
$$D := \operatorname{diag}\left(\hat{S}_{\text{diag}}\right), \qquad \hat{S}_{\text{diag}} := -B\operatorname{diag}(F)^{-1}B^T - C,$$
$$\alpha := \frac{-1}{\rho_{\text{eig}}\left(\hat{S}_{\text{diag},C=0}D^{-1}\right)}, \qquad \hat{S}_{\text{diag},C=0} := -B\operatorname{diag}(F)^{-1}B^T.$$

As in [12, Section 3.3.3], we replace $BM_u^{-1}B^T + \gamma C$ with A_p with no special consideration for the stabilization matrix C:

$$S^{-1}_{\text{LSC}_{\text{stab},A_p}} = -A_p^{-1}(BM_u^{-1}FM_u^{-1}B^T)A_p^{-1} - \alpha D^{-1}.$$

Inverse of the velocity mass matrix. Similar to a strategy applied in the construction of the SIMPLE preconditioner (see Section 3.3), we approximate the inverse of M_u with one of the following two matrices: Let δ^{ij} denote the Kronecker delta. Then

$$H^D_{M_u} := \operatorname{diag}(M_u)^{-1}, \qquad H^{\Sigma}_{M_u} := \delta^{ij}\Big(\sum_{k=1}^{N_u}|(M_u)_{i,k}|\Big)^{-1}.$$

3.3. SIMPLE. The SIMPLE (Semi-Implicit Method for Pressure-Linked Equations) preconditioner was originally introduced by Patankar and Spalding in [67] to solve the Navier–Stokes equations, and the SIMPLE block preconditioner is based on it; see also [26, 71].

First, the lower-triangular and the block-diagonal component of the LDU decomposition (3.1) are grouped together in an LU decomposition:

$$\begin{bmatrix} F & 0 \\ B & S \end{bmatrix}\begin{bmatrix} I & F^{-1}B^T \\ 0 & I \end{bmatrix}.$$

Then, the inverse of F in the upper-right block is replaced by an approximation H_F, which differs depending on whether SIMPLE or SIMPLEC is used; a SIMPLE variant that is not covered in this work is SIMPLER [21, 64]. The Schur complement S is approximated with the matrix $S_{\text{SIMPLE}} = -C - BH_FB^T$. The introduction of an under-relaxation parameter α completes the construction of the preconditioner $\mathcal{B}^{-1}_{\text{SIMPLE}}$:

$$\mathcal{B}_{\text{SIMPLE}} = \begin{bmatrix} F & 0 \\ B & S_{\text{SIMPLE}} \end{bmatrix}\begin{bmatrix} I & \frac{1}{\alpha}H_FB^T \\ 0 & \frac{1}{\alpha}I \end{bmatrix}.$$

H_F is a diagonal matrix and depends on the specific SIMPLE variant. We denote the matrix of the default variant SIMPLE as H_F^D and the one of SIMPLEC as H_F^{Σ}:

(SIMPLE) $$H_F^D = \operatorname{diag}(F)^{-1},$$

(SIMPLEC) $$H_F^{\Sigma} = \delta^{ij}\Big(\sum_{k=1}^{N_u}|F_{i,k}|\Big)^{-1}.$$

If we expand F into its parts

$$F = \frac{3}{2\Delta t}M_u + \nu A + N + W,$$

with the velocity mass matrix M_u, the discrete Laplacian $-A$, and the advective terms $N+W$, then the well-conditioned mass matrix dominates F for small time steps Δt. Since M_u is spectrally equivalent to the lumped mass matrix $\operatorname{diag}(M_u)$, approximating F via its diagonal is sufficient for small Δt. The performance of SIMPLE and SIMPLEC can deteriorate for stationary flow problems if Δt is large or if advective forces dominate.

ETNA
Kent State University and
Johann Radon Institute (RICAM)

4. The additive overlapping Schwarz preconditioners. As discussed in Sections 2.1 and 2.2, we solve the Navier–Stokes equations using a Newton–Krylov approach, that is, by solving the linearized systems arising from Newton's method using a Krylov method. Since the tangent matrix (2.5) is not positive definite and generally unsymmetric, we employ the preconditioned GMRES method [73] as the Krylov method. In Section 3, we have discussed several options to block-precondition the system matrix (2.5), where each application of a block-preconditioner requires the separate solution of velocity and pressure systems. For large-scale problems, it is often unfeasible to use direct solvers for the solution of these subsystems. In this paper, we approximate the involved inverses with overlapping Schwarz domain decomposition methods. Specifically, we use block Schwarz preconditioners and monolithic Schwarz preconditioners, the latter of which are constructed by directly employing Schwarz preconditioners on the system matrix; see Sections 4.2 and 4.3 for details on the block and monolithic approaches, respectively. For the implementation of both techniques, we employ the `Trilinos` package `FROSch`; cf. Section 5.2. We introduce Schwarz preconditioners under simplified assumptions, that is, for a linear system

$$Kx = b,$$

resulting from the discretization of a Laplacian model problem on the computational domain Ω using, for example, piecewise linear finite elements with the finite element space $V^h = V^h(\Omega)$. In this case, the matrix K is symmetric positive definite, and theoretical results for convergence are available. We refer to [77] for a more detailed introduction. The following construction is essentially the same for general problems but is not based on a robust theoretical background.

Let Ω be decomposed into nonoverlapping subdomains $\{\Omega_i\}_{i=1}^N$ that are conforming with the finite element discretization; that is, the closure of every subdomain is the union of its finite elements. We extend the subdomains by k layers of finite elements, resulting in an overlapping domain decomposition $\{\Omega_i'\}_{i=1}^N$ with overlap $\delta = kh$. Note that extending the subdomains by layers of elements is a geometric operation based on the mesh. To only use information available via the matrix K, the subdomains can be algebraically extended by iteratively adding those degrees of freedom that are in the neighborhood of the (overlapping) subdomain, defining neighborhood via the sparsity pattern of K; cf. [42, 44]. For example, if Ω_i is extended by one layer, then, to obtain Ω_i', for all degrees of freedom l in Ω_i, degrees of freedom j for which $K_{l,j} \neq 0$ are added to Ω_i. This algebraic overlap construction is used in our implementation. We denote the size of the algebraically determined overlap by $\hat{\delta} = k$ for k iterations of the overlap-constructing algorithm. Note that in the context of conforming Lagrange finite elements, $\hat{\delta} = k$ amounts to a geometric overlap of $\delta = (k+1)h$.

Based on the overlapping domain decomposition, we define a restriction operator $R_i : V^h \to V_i^h$, $i = 1, \dots, N$, to map from the global finite element space V^h to the local finite element space $V_i^h := V^h(\Omega_i')$ on the overlapping subdomains Ω_i'. Moreover, we define the corresponding prolongation operator $R_i^T : V_i^h \to V^h$, which is the transpose of the restriction operator and extends a local function on Ω_i' by zero outside of Ω_i'. We define a second prolongation $\widetilde{R}_i^T$: The standard additive Schwarz method is given by $\widetilde{R}_i^T = R_i^T$. The restricted additive Schwarz method [10] uses R_i^T but sets values to zero such that $\sum_{i=1}^N \widetilde{R}_i^T R_i$ is the identity operator. In this work, we define $\widetilde{R}_i^T$ to satisfy the identity-operator property via an inverse multiplicity: A row of R_i^T corresponding to the finite element node x^h is scaled with the inverse of $|\{j \in \{1, \dots, N\} : x^h \in \Omega_j' \setminus \partial\Omega_j'\}|$, where $|\cdot|$ is the cardinality of the set; cf. [10, Remark 2.7] and [41, Section 4.1].

Using the aforementioned operators R_i, R_i^T, and $\widetilde{R}_i^T$ and defining local overlapping stiffness matrices $K_i := R_i K R_i^T$, $i = 1, \dots, N$, the (scaled) additive one-level overlapping

ETNA
Kent State University and
Johann Radon Institute (RICAM)

Schwarz preconditioner is given by

$$M_{\text{OS-1}}^{-1} = \sum_{i=1}^{N} \widetilde{R}_i^T K_i^{-1} R_i. \tag{4.1}$$

Even for a scalar elliptic Laplacian model problem, the one-level Schwarz preconditioner (4.1) is generally not numerically scalable: The number of Krylov iterations will increase with an increasing number of subdomains. In order to define a numerically scalable overlapping Schwarz preconditioner that yields a convergence rate independent of the number of subdomains, a coarse level can be introduced. With a coarse interpolation operator $\Phi : V_0 \to V^h$, which is still to be specified, the additive two-level overlapping Schwarz preconditioner reads

$$M_{\text{OS-2}}^{-1} = \Phi K_0^{-1} \Phi^T + \underbrace{\sum_{i=1}^{N} \widetilde{R}_i^T K_i^{-1} R_i}_{=M_{\text{OS-1}}^{-1}}, \tag{4.2}$$

where $K_0 = \Phi^T K \Phi$. The columns of Φ form the basis of the coarse space V_0, and K_0 is a Galerkin projection of K into the coarse space. A natural choice for coarse basis functions is to use Lagrangian basis functions defined on a coarse triangulation, which yields a scalable preconditioner. However, the definition of a coarse triangulation heavily depends on geometric information and cannot be easily done in a conforming way for complex domain geometries. In the next section, we will discuss alternatives.

4.1. GDSW-type coarse spaces. For this work, we choose a very flexible approach for the construction of a coarse space that was introduced with the Generalized Dryja–Smith–Widlund (GDSW) preconditioner in [17, 18]. It is inspired by nonoverlapping domain decomposition methods such as those from FETI-DP [32, 33] and BDDC [11, 15]. The approach allows one to define many different variants, three of which (GDSW, RGDSW, GDSW*) we will introduce and use for simulations. They require little to no additional information to the system matrix. In this approach, the nonoverlapping subdomains $\{\Omega_i\}_{i=1}^N$ are chosen as the elements of the coarse triangulation, and the coarse basis functions are defined via extensions of trace functions defined on the interface

$$\Gamma = \left\{ x \in \left(\Omega_i \cap \Omega_j\right) \setminus \partial\Omega_D : i \neq j, 1 \leq i, j \leq N \right\},$$

where $\partial\Omega_D$ is the global Dirichlet boundary; note the comment in Section 4.1.3 for discontinuous finite elements, in which case the interface does not contain any degrees of freedom. More precisely, first, a partition of unity is defined on Γ. The different variants of the coarse spaces differ only in the choice of the partition of unity on the interface. Then the functions are extended into the interior of the subdomains. The extension is defined algebraically using submatrices of K and corresponds to a discrete harmonic extension in the case of a Laplace problem. In this specific case, the extension of a partition of unity on the interface gives a partition of unity on the whole domain Ω. Given interface values Φ_Γ, the full matrix Φ is obtained by solving for all degrees of freedom I that do not correspond to the interface Γ:

$$\Phi = \begin{bmatrix} \Phi_I \\ \Phi_\Gamma \end{bmatrix} = \begin{bmatrix} -K_{II}^{-1} K_{I\Gamma} \Phi_\Gamma \\ \Phi_\Gamma \end{bmatrix}. \tag{4.3}$$

The blocks K_{II} and $K_{I\Gamma}$ are obtained by a reordering and partitioning of K as

$$K = \begin{bmatrix} K_{II} & K_{I\Gamma} \\ K_{\Gamma I} & K_{\Gamma\Gamma} \end{bmatrix}.$$

ETNA
Kent State University and
Johann Radon Institute (RICAM)

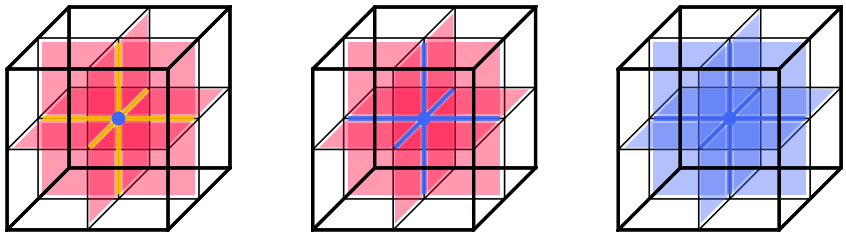

FIG. 4.1. Visualization of three types of (overlapping) interface decompositions in a $2 \times 2 \times 2$ domain decomposition. *Left: GDSW interface components: 6 edges, 1 vertex, and 12 faces. Center: GDSW* interface components: 1 vertex-based component (union of vertex and 6 edges), 12 faces. Right: RGDSW interface component: 1 vertex-based component (union of vertex, 6 edges, 12 faces).*

Note that K_{II} is block-diagonal, and hence the solver required for K_{II}^{-1} can be dealt with independently for each subdomain. Moreover, as only submatrices of K are required, the construction of the coarse space is algebraic. For other problems, like elasticity, additional geometric information may be required; cf., e.g., [19, Section 4]. In this work, for the fluid problems, we do not make use of geometric information and obtain an algebraically constructed coarse space. The description above outlines the construction for a scalar problem; the extension to vector-valued problems is given in Section 4.1.2. The coarse spaces used in this work are all based on GDSW-type preconditioners and on extending the interface values into the interior via (4.3); they only differ in the definition of the interface partition of unity Φ_Γ. We review the different approaches implemented in `FROSch` in Section 6. We consider three coarse-space variants: GDSW [17, 18] and reduced-dimension GDSW (RGDSW) [20] ("Option 1")—due to its significantly reduced dimension, the RGDSW coarse space improves parallel scalability; cf. [46]—and we will introduce an intermediate approach that we denote as GDSW*. It yields smaller coarse spaces than the classical GDSW approach but larger coarse spaces than the RGDSW approach; in two dimensions, GDSW* and RGDSW are identical. Preliminary results for GDSW* based on its implementation in `FROSch` can be found in [50, p. 160], there under the working title *GDSW-Star*. A related approach was taken in [60] in the context of adaptive coarse spaces, there under the name R-WB-GDSW.

4.1.1. Interface partition of unity of GDSW, GDSW*, and RGDSW coarse spaces. The classical GDSW coarse space defines the interface partition of unity based on a nonoverlapping decomposition of the interface degrees of freedom into interface components of faces, edges, and vertices; cf. Figure 4.1. By contrast, the RGDSW coarse spaces from [20] employ overlapping interface components, combining each vertex with all its adjacent faces and edges; the resulting decomposition overlaps in the faces and edges. Combining vertices with edges and faces reduces the number of interface components from the number of faces, edges, and vertices to only the number of vertices. In [43], a variant of the RGDSW coarse space is introduced in which the RGDSW interface decomposition is made nonoverlapping without increasing the number of interface components. The new GDSW* coarse space employs a partition of unity of intermediate size, where each vertex is only combined with the adjacent edges and the faces are kept separate.

The interface partition of unity is obtained by defining one function for each interface component. As a result, the number of coarse basis functions, which is equal to the coarse-space dimension, is reduced for GDSW* and even further reduced for RGDSW; cf. Table 4.1. See also [60, Figure 5.4] for a comparison of the number of interface components for four different types of meshes and domain decompositions; note that, although the precise definition of the interface components differs in [60], their quantity is the same. There, the analogue of

ETNA
Kent State University and
Johann Radon Institute (RICAM)

TABLE 4.1
Coarse-space dimensions for the scalar Laplace problem and the different GDSW-type coarse spaces used in this work. *Vertices, edges, and faces refer to the respective interface components of the domain decomposition.*

Coarse Space	Coarse Space Dimension
GDSW	number of vertices + faces + edges
GDSW*	number of vertices + faces
RGDSW	number of vertices

GDSW* is denoted as R-WB-GDSW.

To define the interface partition of unity of the GDSW, GDSW*, and RGDSW coarse spaces, we decompose the interface Γ into components $\gamma_k \subset \Gamma$ such that

$$\Gamma = \bigcup_k \gamma_k.$$

This union can be disjoint or overlapping. Then, we define functions $\Phi_{\Gamma,k}$ on Γ, with $\operatorname{supp}(\Phi_{\Gamma,k}) \subset \gamma_k$, such that

$$\sum_k \Phi_{\Gamma,k} \equiv 1 \quad \text{on } \Gamma.$$

Then, we gather the functions $\Phi_{\Gamma,k}$ (that is, their respective coefficient vectors) as columns of a matrix

$$\Phi_\Gamma = \begin{bmatrix} \Phi_{\Gamma,1} & \Phi_{\Gamma,2} & \dots \end{bmatrix},$$

which is subsequently extended using (4.3).

It remains to define the partition-of-unity functions $\Phi_{\Gamma,k}$ for GDSW, GDSW*, and RGDSW. We introduce the following formal definitions of faces, edges, and vertices. In three dimensions:

- A face is a set of nodes that belongs to the same two subdomains. Let $\mathcal{F}$ be the set of all faces.
- An edge is a set of at least two nodes that belongs to the same (more than two) subdomains. Let $\mathcal{E}$ be the set of all edges.
- A vertex is a single node that belongs to the same (more than two) subdomains. Let $\mathcal{V}$ be the set of all vertices.

These interface components are exemplarily shown in Figure 4.1 (left); all of them are used to construct GDSW coarse spaces. In two dimensions, the definition of an edge is the same as that of a face in 3D, and the definition of the vertex remains the same. In the following, we introduce the different partitions of unity considered in this work, focusing on the three-dimensional case.

GDSW. For the construction of the coarse space, the γ_k are given by faces, edges, and vertices. We define the discrete partition-of-unity functions via

$$\chi_f, f \in \mathcal{F}, \qquad \chi_e, e \in \mathcal{E}, \qquad \chi_v, v \in \mathcal{V};$$

χ_s is the discrete characteristic function that takes the value one in the finite elements nodes contained in s and zero elsewhere. Then, we have that

$$\sum_{f\in\mathcal{F}} \chi_f + \sum_{e\in\mathcal{E}} \chi_e + \sum_{v\in\mathcal{V}} \chi_v \equiv 1 \qquad \text{on } \Gamma,$$

since the vertices, edges, and faces form a nonoverlapping decomposition of Γ.

ETNA
Kent State University and
Johann Radon Institute (RICAM)

RGDSW. Let $\mathcal{V} = \{v_1, \dots, v_{N_\mathcal{V}}\}$. An interface component γ_k is the union of $v_k \in \mathcal{V}$ and its adjacent faces and edges. Note that, as discussed in [20], the definition is slightly more involved for general domain decompositions, where not every edge and face may have an adjacent vertex; for simplicity, we do not discuss these cases.

In [20, 43] various options for defining the corresponding partition-of-unity functions are given. Here, we employ and describe the algebraic variant "Option 1" from [20], which is based on a simple multiplicity scaling (cf. [20, Equation (1)]). In particular, we first define a component multiplicity $\mathcal{M}(n)$ of an interface node n as the number of (RGDSW) interface components γ_k that contain n:

$$\mathcal{M}(n) := |\{k : n \in \gamma_k\}|.$$

For example, for an edge with two adjacent vertices, we have $\mathcal{M}(n) = 2$ for all nodes n on the edge. We can then define a function $\Phi_{\Gamma,k}$ associated with γ_k by using the inverse of the multiplicity:

$$\Phi_{\Gamma,k}(n) := \begin{cases} \frac{1}{\mathcal{M}(n)} & n \in \gamma_k, \\ 0 & \text{else.} \end{cases} \tag{4.4}$$

As the partition-of-unity property is satisfied in every interface node $n \in \Gamma$, we have

$$\sum_k \Phi_{\Gamma,k}(n) = 1.$$

GDSW⋆. Finally, we introduce the partition of unity of the new GDSW⋆ coarse space. The definition is based directly on those of GDSW and RGDSW. In particular, we define two types of interface components:

1. A vertex $v_k \in \mathcal{V}$ and its adjacent edges. These components overlap.
2. A face $f_k \in \mathcal{F}$. Faces do not overlap with the vertex-based interface components or other faces.

We define the partition of unity using the inverse multiplicity scaling (4.4), this time defined via the set of GDSW⋆ interface components. Note that, because the faces do not overlap with other interface components, the resulting functions coincide with the discrete characteristic functions χ_f of the faces $v \in \mathcal{F}$.

As faces do not exist in two dimensions, the partition of unity of GDSW⋆ and RGDSW are the same in this case.

4.1.2. Extension to vector-valued problems. The approaches discussed before are directly applicable to scalar problems such as problems related to the pressure block. However, the application to vector-valued problems, such as problems related to the velocity block, requires a small modification.

In the case of, for example, a multidimensional Laplace operator $\Delta\mathbf{u}$ or a linearized elasticity problem, the partition-of-unity functions γ_k are multiplied with a basis of the null space of the corresponding partial differential equation with homogeneous Neumann conditions on the boundary. This ensures that the null space can be represented by the coarse space, which is required for numerical scalability in case of symmetric positive semidefinite problems; cf. [76, p. 132]. In our case, we use translations for the velocity block, which is motivated by $\Delta\mathbf{u}$ appearing in the diagonal velocity block of the discrete Stokes matrix. For the pressure, we use constant functions; note that these are in the null space of B^T for an enclosed-flow problem.

ETNA
Kent State University and
Johann Radon Institute (RICAM)

Specifically, for the two-dimensional case of a velocity coarse function, the partition-of-unity functions are multiplied with

$$r_1 := \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \qquad r_2 := \begin{bmatrix} 0 \\ 1 \end{bmatrix},$$

and in three dimensions with

$$r_1 := \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \qquad r_2 := \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \qquad r_3 := \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix},$$

for each interface node. As a result, the number of coarse space functions associated with the velocity or pressure is the space dimension multiplied with the number of partition-of-unity functions. This discussion is compatible with the scalar elliptic case. In particular, for the construction of pressure functions, the nodewise multiplication of the partition-of-unity functions with the scalar constant function, i.e.,

$$r := \begin{bmatrix} 1 \end{bmatrix},$$

simply yields the partition-of-unity functions.

4.1.3. Discontinuous finite elements. For discontinuous finite element spaces, as employed in discontinuous, piecewise linear (P1-disc.) pressure spaces, there are no interface nodes that are shared between adjacent subdomains. Therefore, the discrete interface is empty in this case, and the construction of GDSW, GDSW*, and RGDSW cannot be applied directly. As motivated in Section 4.1.2, we want to use constant functions for the pressure. Therefore, on each subdomain Ω_i, we employ the single discrete characteristic function χ_{Ω_i} (i.e., a function that is constant on Ω_i and zero elsewhere) as the coarse basis function, yielding a total of N coarse basis functions. If this occurs, the coarse space will be abbreviated as "Vol." since the characteristic function is prescribed on the entire volume and not just on the interface.

4.2. The block Schwarz preconditioner. For the presented block preconditioners in Section 3, we can use the two-level overlapping Schwarz domain decomposition method from the previous section for the approximation of the blocks F^{-1} and S^{-1}. The inverse of the Schur complement is not approximated directly; for example, in the case of $S_{\text{PCD}}^{-1} = -A_p^{-1} F_p M_p^{-1}$, we may approximate A_p^{-1} and M_p^{-1} with a Schwarz method. Nevertheless, to avoid new notation, we assume in the following that S^{-1} shall be approximated. We define the overlapping subdomain problems separately for the velocity and pressure. The velocity and pressure space $\mathbf{V}^h$ and Q^h, respectively, are decomposed into the local spaces

$$\mathbf{V}_i^h = \mathbf{V}^h \cap (H_0^1(\Omega_i'))^3 \qquad \text{and} \qquad Q_i^h = Q^h \cap H_0^1(\Omega_i'),$$

for $i = 1, \ldots, N$, on the overlapping subdomains Ω_i'. Note that here we restrict ourselves to conforming finite element spaces; the precise definition for Q2-P1-disc. is different for the discontinuous pressure space since H^1 cannot be used and since functions of the local finite element spaces with a zero boundary are not only zero at the boundary nodes of Ω_i' but also on the element layer next to the boundary of Ω_i'. The operators restricting to the overlapping subdomains belonging to the velocity and pressure degrees of freedom are defined as

$$R_{u,i} : \mathbf{V}^h \longrightarrow \mathbf{V}_i^h \qquad \text{and} \qquad R_{p,i} : Q^h \longrightarrow Q_i^h,$$

respectively. Based on this, the local overlapping problems are defined as

$$F_i = R_{u,i} F R_{u,i}^T \qquad \text{and} \qquad S_i = R_{p,i} S R_{p,i}^T,$$

ETNA
Kent State University and
Johann Radon Institute (RICAM)

where S, or an approximation of S, depends on the underlying strategy. The coarse matrix is constructed similarly to before via a Galerkin product as

$$F_0 = \Phi_u^T F \Phi_u \qquad \text{and} \qquad S_0 = \Phi_p^T S \Phi_p.$$

Let $\widetilde{R}_{u,i}^T$ and $\widetilde{R}_{p,i}^T$ be the scaled prolongation operators (cf. Section 4). To approximate the inverses of F and S, we use the additive two-level overlapping Schwarz preconditioners

$$\hat{F}^{-1} = \Phi_u F_0^{-1} \Phi_u^T + \sum_{i=1}^{N} \widetilde{R}_{u,i}^T F_i^{-1} R_{u,i} \quad \text{and} \quad \hat{S}^{-1} = \Phi_p S_0^{-1} \Phi_p^T + \sum_{i=1}^{N} \widetilde{R}_{p,i}^T S_i^{-1} R_{p,i}.$$

4.3. The monolithic Schwarz preconditioner. For the saddle-point system with system matrix $\mathcal{F}$ defined in (2.5), we compare the block Schwarz preconditioners with monolithic Schwarz preconditioners. This technique has been introduced in [57, 58] with Lagrangian coarse spaces and extended to GDSW-type coarse spaces in [40, 41]. In monolithic Schwarz preconditioners, we construct a single Schwarz domain decomposition preconditioner for the full system $\mathcal{F}$ at once. The monolithic additive two-level Schwarz preconditioner reads

$$\mathcal{M}_{\text{OS-2}}^{-1} = \phi \mathcal{F}_0^{-1} \phi^T + \sum_{i=1}^{N} \widetilde{\mathcal{R}}_i^T \mathcal{F}_i^{-1} \mathcal{R}_i. \tag{4.5}$$

Different from (4.2), we denote all matrices with calligraphic letters to indicate that they are block matrices with the same block structure as $\mathcal{F}$. The local subdomain matrices for (4.5) are given by $\mathcal{F}_i = \mathcal{R}_i \mathcal{F} \mathcal{R}_i^T$ with the restriction operators

$$\mathcal{R}_i = \begin{bmatrix} R_{u,i} & 0 \\ 0 & R_{p,i} \end{bmatrix},$$

where $R_{u,i}$ and $R_{p,i}$ correspond to the restriction operators for the velocity $\mathbf{u}$ and pressure p degrees of freedom for Ω_i'. $\widetilde{\mathcal{R}}_i^T$ is the scaled prolongation operator; cf. Section 4. Then, also the coarse matrix $\mathcal{F}_0 = \phi^T \mathcal{F} \phi$ has a block structure given by the matrix

$$\phi = \begin{bmatrix} \phi_{\mathbf{u},\mathbf{u_0}} & \phi_{\mathbf{u},p_0} \\ \phi_{p,\mathbf{u_0}} & \phi_{p,p_0} \end{bmatrix}, \tag{4.6}$$

which has the coarse basis functions as its columns. $\mathbf{u_0}$ and p_0 indicate the velocity and pressure functions in the coarse space. The interface values of the coarse functions are constructed as for the block matrices; the remaining interface values are always set to zero. Hence, we obtain the structure

$$\phi_\Gamma = \begin{bmatrix} \phi_{\Gamma,\mathbf{u},\mathbf{u_0}} & 0 \\ 0 & \phi_{\Gamma,p,p_0} \end{bmatrix}.$$

For a block Schwarz preconditioner, the same $\phi_{\Gamma,\mathbf{u},\mathbf{u_0}}$ and ϕ_{Γ,p,p_0} are used. Only the subsequent extension operator from the interface to the subdomains differs. The extension from the interface to the subdomains is computed monolithically, analogously to (4.3) but with submatrices of $\mathcal{F}$. As a result, the coarse basis ϕ also contains components reflecting the coupling between $\mathbf{u}$ and p; cf. (4.6).

Since the local saddle-point subdomain matrices $\mathcal{F}_i$ are extracted from $\mathcal{F}$, they possess homogeneous Dirichlet boundary conditions for both velocity and pressure in the case of conforming finite elements. Additionally, we enforce a zero-mean pressure value on the

ETNA
Kent State University and
Johann Radon Institute (RICAM)

local overlapping subdomains as in *Version 2* in [57, Section 4.1], which is beneficial in some cases, even though the pressure is uniquely determined by the problem setting; see also [40, Section 5.1]. To this end, we introduce a pressure projection in the first level of the two-level overlapping Schwarz preconditioner

$$\mathcal{M}_{\text{OS-2}}^{-1} = \phi \mathcal{F}_0^{-1} \phi^T + \sum_{i=1}^{N} \widetilde{\mathcal{R}}_i^T \overline{\mathcal{P}}_i \mathcal{F}_i^{-1} \mathcal{R}_i \tag{4.7}$$

with local projection operators $\overline{\mathcal{P}}_i$ corresponding to the overlapping subdomain Ω_i. Here, the projection operators are of the form

$$\overline{\mathcal{P}}_i = \begin{bmatrix} I_{u,i} & 0 \\ 0 & \overline{P}_{p,i} \end{bmatrix}, \qquad \overline{P}_{p,i} = I_{p,i} - a_i (a_i^T a_i)^{-1} a_i^T.$$

The vector a_i is defined as $a_i = R_{p,i} a_p$ via integrals of the pressure basis functions ψ_j:

$$a = \begin{bmatrix} 0 \\ a_p \end{bmatrix}, \qquad a_p = \begin{bmatrix} \int_\Omega \psi_1 \, \mathrm{d}x & \dots & \int_\Omega \psi_{N_p} \, \mathrm{d}x \end{bmatrix}^T.$$

Results in [40, 57] show that this approach, similar to introducing a Lagrange multiplier, can improve the convergence of the iterative linear solver.

5. Implementation. The software framework of this paper is based on the `Trilinos` library; in particular, we employ its domain decomposition package `FROSch` (Fast and Robust Overlapping Schwarz) [44] and its block-preconditioning package `Teko` [12, 13] to construct the preconditioners under investigation. Moreover, for the implementation of the model problems and the corresponding finite element discretizations, we use the `FEDDLib` (Finite Element and Domain Decomposition Library) [34], which strongly relies on several `Trilinos` packages and assembles input directly in a format to make use of the full capabilities of `FROSch` and `Teko`.

We have carried out some minor modifications to `Teko` in `Trilinos`[1] for the use of PCD and LSC in the `FEDDLib`. Furthermore, the implementation of the pressure projection is not yet contained in the official `FROSch` repository.

5.1. Trilinos and FEDDLib. The `Trilinos` library [78] is a collection of interoperable software packages for high-performance scientific computing. Since the list of packages is extensive, we will focus on those of relevance to our paper. The main building block for the preconditioners investigated in this paper is the parallel linear algebra package `Tpetra`, which provides parallel vector and matrix classes as well as functions that allow communication of those objects. It entails a software stack of packages that operate on `Tpetra` objects such as direct and iterative solvers and preconditioners. `Tpetra` automatically incorporates access to the `Kokkos` performance portability framework [22, 80] and the corresponding kernel library `Kokkos Kernels` [72]. As a result, functions of `Tpetra` objects allow for node-level parallelization on CPUs and GPUs via `Kokkos`' node-type template parameter; here, we only focus on distributed memory parallelization, and hence, we will not consider the effects of node-level parallelization via `Kokkos`. The older `Epetra` linear algebra package is currently being deprecated together with the whole `Epetra`-only stack of packages. Therefore, even though our software framework supports the `Epetra` stack, we employ only the `Tpetra` stack.

[1] Our implementation and results in this paper are based on `Trilinos` in the version with commit ID 23ccc58 (master branch); see the GitHub repository [78].

ETNA
Kent State University and
Johann Radon Institute (RICAM)

The model problems are implemented and assembled using the FEDDLib [34]. Currently, it is split into three main packages, amr, core, and problems. The amr package is for adaptive mesh refinement, which is not employed in this work. The core package includes the general implementation of the finite element assembly, IO tools, the interface to the Tpetra parallel linear algebra, mesh classes, and further smaller utility functions. The problem package provides implementations of different physics problems supported by FEDDLib, including the computational fluid dynamics problems considered in this work. Further problems can be added easily using the interface to the Mathematica code generation package AceGen [61]; the interface is discussed in more detail in [3]. Furthermore, FEDDLib's problem package includes interfaces to the nonlinear solver package NOX, which will also be used for solving the nonlinear problems in this work, and the unified solver interface Stratimikos.

Stratimikos enables the use of iterative solvers implemented in Belos and of interfaces for direct solvers, which are accessible via the Amesos2 package. In order to accelerate the convergence of iterative solvers, Trilinos provides various preconditioners, including:

- Ifpack2: one-level Schwarz preconditioners, supporting incomplete factorizations on the subdomains;
- Teko: block preconditioners;
- FROSch: multi-level Schwarz preconditioners;
- MueLu: algebraic multigrid, as a solver or preconditioner.

As mentioned before, we focus on Teko and FROSch preconditioners; see Sections 3 and 4, respectively, for details on the preconditioners employed in this work. Details on Teko are given in Section 5.3 and on FROSch in Section 5.2.

5.2. Fast and Robust Overlapping Schwarz (FROSch). FROSch (Fast and Robust Overlapping Schwarz) [44, 45] is a parallel domain decomposition preconditioning package within the Trilinos software library [49, 65]. It builds on the Schwarz framework [77] to construct preconditioners by combining elementary Schwarz operators, for instance, implementing the two-level Schwarz preconditioner by combining the first- and second-level operators; cf. (4.1) and (4.2).

A key feature of FROSch is its use of extension-based coarse spaces that avoid explicit geometric information for the problems of this paper (cf. Section 4.1) and, in particular, the need for coarse triangulations; see Section 4 for a detailed discussion of the methodology. The coarse spaces are based on the GDSW preconditioner [17, 18] and have been adapted for block systems [40]. Moreover, FROSch also implements the reduced-dimension RGDSW variants [46]. Multilevel extensions [47] further enhance scalability, enabling FROSch to scale to over 220 000 cores [48].

FROSch integrates with Trilinos' unified solver interface Stratimikos and supports both parallel linear algebra frameworks, Epetra and Tpetra, via the lightweight Xpetra interface; due to the recent deprecation of Epetra, its support is merely a legacy feature. The overlapping domain decomposition is constructed algebraically using the sparsity pattern of the parallel distributed input matrix. With Tpetra, FROSch leverages the performance portability of Kokkos and Kokkos Kernels for efficient parallelization on CPUs and GPUs [82].

FROSch preconditioners are compatible with iterative Krylov solvers from the Belos package [4], either directly or through Stratimikos. For more details, see [40, 44, 45].

5.3. The block preconditioners in Teko. Teko is a Trilinos package for block-preconditioning multiphysics problems. It can be used to build generic preconditioning strategies for arbitrary problems, the simplest of which is a diagonal preconditioner. Specif-

ETNA
Kent State University and
Johann Radon Institute (RICAM)

LISTING 1
Pressure Convection-Diffusion (PCD): parameter XML file for `Teko` *(*`Trilinos`*).*

```
<ParameterList name="PCD">
  <Parameter name="Type" type="string" value="Block LU2x2"/>
  <!-- Using LDU decomposition for preconditioning structure (true) or upper
    block-triangular (false) -->
  <Parameter name="Use LDU" type="bool" value="false"/>
  <Parameter name="Strategy Name" type="string" value="NS PCD Strategy"/>
  <ParameterList name="Strategy Settings">
    <Parameter name="PCD Operator" type="bool" value="true"/>
    <Parameter name="Pressure Laplace Operator" type="bool" value="true"/>
    <Parameter name="Pressure Mass Operator" type="bool" value="true"/>
    <!--FROSch preconditioner for approximation of inverses -->
    <Parameter name="Inverse F Type" type="string" value="FROSchVelocity"/>
    <Parameter name="Inverse Laplace Type" type="string"
    value="FROSchPressure"/>
    <!-- Approximation of inverse of pressure mass matrix: FROSch
    preconditioner (FROSchPressure) or diagonal approximation (Diagonal,
    AbsRowSum) -->
    <Parameter name="Inverse Mass Type" type="string"
    value="FROSchPressure"/>
    <Parameter name="Flip Schur Complement Ordering" type="bool"
    value="true"/>
  </ParameterList>
</ParameterList>
```

ically, for the Navier–Stokes problem there exist a number of block preconditioners within `Teko`. We consider the block preconditioners presented in Section 3 and different modifications that can be applied. The SIMPLE, SIMPLEC, LSC, and PCD preconditioners are part of the `NS` (Navier–Stokes) namespace within `Teko`. For a detailed description of `Teko` and the Navier–Stokes preconditioners within `Teko`, we refer to [12, 13].

The SIMPLE and SIMPLEC preconditioner can be constructed algebraically based on the underlying system to be solved, without the assembly of additional matrices. For LSC and PCD, to make components of the Schur complement that require additional construction available to `Teko`, callback functions are defined, e.g., for A_p and F_p of the PCD preconditioner. These operators are constructed outside of `Teko`, in our case in the `FEDDLib`. Only the callback function for the velocity mass matrix is optional; if it is not provided, then a diagonal matrix based on F is substituted as an approximation.

For the approximations $\hat{F}^{-1}$ and $\hat{S}^{-1}$ of the inverses of the fluid matrix and Schur complement, respectively, the specification of inverse types is necessary. If no inverse type is specified, then the default is to use a direct solver from the `Amesos2` package. Ideally, a less expensive approximation strategy is specified. For our simulations, we use different two-level overlapping Schwarz preconditioners from the `Trilinos` package `FROSch`; see Table 5.2 for an overview of the used combinations of preconditioning strategies.

5.3.1. The PCD block preconditioner. The PCD preconditioner in `Teko` is derived from the abstract class `LU2x2Strategy`. To use it, the parameter `Type` is set to *Block LU2x2* and the `Strategy Name` to *NS PCD Strategy*; see Listing 1. For the LSC and SIMPLE preconditioners, the `Type` directly refers to the block preconditioner, e.g., it is set to *NS LSC* instead of *LU2x2Strategy*; see the Listings 2 and 3 for the LSC and SIMPLE parameter files.

ETNA
Kent State University and
Johann Radon Institute (RICAM)

TABLE 5.1
Comparison of the block-triangular and the LDU approach for the PCD preconditioner. *Stationary simulation of backward-facing step (see Section 6.1.1) with P2-P1 discretization, $H/h = 9$, and 1 125 cores. PCD block preconditioner with boundary condition type (BC-2) and RGDSW-RGDSW coarse space. Required number of nonlinear steps in parentheses. See Section 6 for parameters of the nonlinear and linear solver. Total time consists of setup and solve time.*

	Full LDU	Block-triangular
# avg. iterations	**78** (5)	103 (5)
Setup time	14.0 s	14.4 s
Solve time	77.1 s	56.9 s
Total	91.1 s	**71.3** s

For PCD, the order of the factors within the Schur complement approximation can be flipped by specifying the parameter `Flip Schur Complement Ordering`:

- $S_{\text{PCD}}^{-1} = -M_p^{-1} F_p A_p^{-1}$, specified with *false* ("default"),
- $S_{\text{PCD}}^{-1} = -A_p^{-1} F_p M_p^{-1}$, specified with *true* ("flipped"; cf. (3.7)).

The "default" Schur complement ordering corresponds to an earlier version of the Schur complement approximation based on a commutator that uses a gradient operator for the pressure instead of a divergence operator for the velocity; see in [31] the equations (2.8) for the "default" and (2.2) for the "flipped" version; see also [29, Remark 9.3].

The Schur complement is constructed and defined as a linear operator. To provide all required operators for the PCD preconditioner, callback functions are defined for M_p, A_p, and F_p. `Teko` then creates approximations for M_p^{-1} and A_p^{-1} according to the parameter file. The inverse of A_p is approximated by an overlapping Schwarz method via `FROSch`. This is indicated by `Inverse Laplace Type` set to `FROSchPressure`, where this value refers to another[2] parameter list that contains the settings for `FROSch`. The inverse of M_p will either be approximated via `FROSch` or by using the same approach as in LSC and SIMPLE for M_u and F, respectively; that is, we can use a diagonal approximation of M_p^{-1},

$$H_{M_p}^{D} = \operatorname{diag}(M_p)^{-1}, \qquad H_{M_p}^{\Sigma} = \delta^{ij} \Big(\sum_{k=1}^{N_p} |(M_p)_{i,k}| \Big)^{-1},$$

referred to as *Diagonal* and *AbsRowSum* scaling, respectively. Note that the pressure convection-diffusion operator needs to be updated in each Newton or fixed-point iteration, while A_p and M_p do not change.

The PCD preconditioner introduced in Section 3.1 is a block-triangular preconditioner (3.2). In `Teko`, the PCD preconditioner can be applied as a full LDU block preconditioner (based on (3.1)) or as a block-triangular preconditioner. The results in Table 5.1 show, compared to the block-triangular preconditioner, how the LDU solve can improve the iteration count, but the total time is larger as its application is more costly. Consequently, we use the classic block-triangular approach:

1. $p = \hat{S}_{\text{PCD}}^{-1} f_p$,
2. $\mathbf{u} = \hat{F}^{-1}(\mathbf{f}_u - B^T p)$.

[2] See, for example, the parameter file `feddlib/problems/examples/unsteadyNavierStokes/parametersTeko.xml` in `FEDDLib`'s GitHub repository [34] (commit ID 1246cdd), where a corresponding entry exists in the sublist `Inverse Factory Library`.

ETNA
Kent State University and
Johann Radon Institute (RICAM)

LISTING 2
Least-Squares Commutator (LSC): Pressure-Laplace Parameter XML file for `Teko` (`Trilinos`).

```
<ParameterList name="LSC-Pressure-Laplace">
  <Parameter name="Type" type="string" value="NS LSC"/>
  <Parameter name="Strategy Name" type="string" value="Pressure Laplace"/>
  <ParameterList name="Strategy Settings">
    <!-- FROSch preconditioner for approximation of inverses -->
    <Parameter name="Inverse Velocity Type" type="string"
    value="FROSchVelocity"/>
    <Parameter name="Inverse Pressure Type" type="string"
    value="FROSchPressure"/>
    <!-- Diagonal approximation type of the inverse of the velocity mass
    matrix -->
    <Parameter name="Scaling Type" type="string" value="AbsRowSum"/>
    <!-- Use of mass matrix (true) or F -->
    <Parameter name="Use Mass Scaling" type="bool" value="true"/>
    <!-- Assuming stable (true) or stabilized  discretization (false) -->
    <Parameter name="Assume Stable Discretization" type="bool" value="true"
    />
  </ParameterList>
</ParameterList>
```

Here, F^{-1} and S_{PCD}^{-1} have been replaced with their approximations based on a Schwarz method, $\hat{F}^{-1}$ and $\hat{S}_{\text{PCD}}^{-1}$; see Table 5.2 for an overview of the approximations used in preconditioners.

As covered in Section 3.1.2, we need to additionally account for the boundary conditions that would naturally arise from the original Schur complement S. The boundary information is integrated into A_p and F_p by the `FEDDLib`. When we set the Dirichlet boundary conditions in the different operators, we zero out the row and keep the diagonal entry; cf. [13, p. S325]. The use of Robin boundary conditions additionally requires the assembly of a matrix on the surface elements of the respective boundary.

5.3.2. The LSC block preconditioner. To use the LSC preconditioner in `Teko`, the parameter `Type` is set to *NS LSC*; see Listing 2. The LSC preconditioner, just like PCD, is an upper block-triangular preconditioner that is applied in two steps:

1. $p = \hat{S}_{\text{LSC}}^{-1} f_p$,
2. $\mathbf{u} = \hat{F}^{-1}(\mathbf{f}_u - B^T p)$.

The Schur complement S_{LSC} is defined in (3.13). Similarly to before, F^{-1} and S_{LSC}^{-1} have been replaced with their approximations $\hat{F}^{-1}$ and $\hat{S}_{\text{LSC}}^{-1}$ based on a Schwarz method. We have different options to treat the inverse of M_u: M_u can be passed along to `Teko` via a callback function. Inverting M_u is expensive, but it can be well approximated with $H_{M_u}^D$ or $H_{M_u}^\Sigma$ (see Section 3.2 for the definition). Via the parameter file, $H_{M_u}^D$ can be selected with *Diagonal* and $H_{M_u}^\Sigma$ with *AbsRowSum*. If M_u is not provided via a callback function, then by default it is replaced with $\text{diag}(F)$. If M_u is replaced with the identity matrix, then this leads to the BFBt preconditioner in (3.14), presented in [30].

Similarly to the PCD preconditioner, we also have the option to replace $BM_u^{-1}B^T$ for continuous pressure elements with the pressure-Laplacian A_p, the resulting Schur complement approximation we denote by $S_{\text{LSC}_{A_p}}$; see (3.15). In that case, we need to account for boundary conditions; we use the same method as in Section 5.3.1 to implement the boundary conditions in A_p and always prescribe a Dirichlet condition on the outlet, which corresponds to the boundary condition strategy (BC-2) of PCD.

ETNA
Kent State University and
Johann Radon Institute (RICAM)

LISTING 3
SIMPLEC: Parameter XML file for `Teko (Trilinos)`.

```
<ParameterList name="SIMPLE">
  <Parameter name="Type" type="string" value="NS SIMPLE"/>
  <!--FROSch preconditioner for approximation of inverses -->
  <Parameter name="Inverse Velocity Type" type="string"
    value="FROSchVelocity"/>
  <Parameter name="Inverse Pressure Type" type="string"
    value="FROSchPressure"/>
  <!-- Definition of diagonal approximation type for H_F: Diagonal (SIMPLE),
    AbsRowSum (SIMPLEC), Lumped, BlkDiag-->
  <Parameter name="Explicit Velocity Inverse Type" type="string"
    value="AbsRowSum"/>
  <!-- Under-relaxation parameter -->
  <Parameter name="Alpha" type="double" value=".9"/>
</ParameterList>
```

The preconditioner is part of the `NS` (Navier–Stokes) subpackage of `Teko` as `InvLSCStrategy` or `PresLaplaceLSCStrategy`, where the *Strategy Name* specifies the strategy (see Listing 2).

To compute the spectral radius of a matrix M in the stabilized variant, `Teko` uses a block Krylov–Schur method with block size 5 and tolerance 0.05 for the convergence criterion $\|Mx - \lambda x\|_2/\|\lambda\|_2$, where x is an approximation of an eigenvector and λ that of the corresponding eigenvalue. This is carried out in each Newton iteration if the preconditioner is rebuilt.

5.3.3. The SIMPLE block preconditioner. The `SIMPLE` preconditioner is also part of the `NS` subpackage of `Teko`. It is selected by setting the parameter `Type` to *NS SIMPLE*; see Listing 3. The type used to approximate the inverse of F determines whether it is called SIMPLE or SIMPLEC: the type *Diagonal* corresponds to SIMPLE and *AbsRowSum* to SIMPLEC. The inverses of F and the Schur complement are replaced with approximations $\hat{F}^{-1}$ and $\hat{S}^{-1}_{\text{SIMPLE}}$ based on a Schwarz method. The under-relaxation parameter is always set to $\alpha = 0.9$; cf. [12, p. 350]. Unlike in the case of PCD and LSC, the application of SIMPLE and SIMPLEC requires three steps:

1. $\mathbf{u}^* = \hat{F}^{-1}\mathbf{f}_u$,
2. $p = \alpha\hat{S}^{-1}_{\text{SIMPLE}}(f_p - B\mathbf{u}^*)$,
3. $\mathbf{u} = \mathbf{u}^* - \frac{1}{\alpha}H_F B^T p$.

5.4. Comparison methodology. In order to compare monolithic and block preconditioning for the Navier–Stokes equations, we evaluate the time to set up the preconditioners, the time taken by the Krylov method (henceforth called "solve" time or cost), and the average iteration count per Newton step. The solve time includes the application of the preconditioner. The PCD and LSC preconditioners are block-triangular preconditioners and, thus, consist of two steps. On the other hand, the SIMPLE preconditioner contains the application of a backward and forward substitution and consists in total of three application steps. Independent of this, the block preconditioners contain approximations of inverses based on a Schwarz method. The monolithic preconditioner contains only the approximation of the inverse of $\mathcal{F}$ based on a Schwarz method. For the monolithic preconditioner, the setup time consists of the approximation of the inverse of $\mathcal{F}$, and for the block preconditioner it consists of the approximation of the inverses of F and the Schur complement (which includes approximations of, e.g., A_p^{-1} using a Schwarz method). Table 3.1 illustrates the different operators that need

ETNA
Kent State University and
Johann Radon Institute (RICAM)

TABLE 5.2
Overview of approximations of inverses used for different preconditioning strategies. $\mathbf{A_S}\left[\cdot\right]$ *denotes the application of an additive Schwarz preconditioner. In the case of a monolithic preconditioner, the entire system* $\mathcal{F}^{-1}$ *is approximated with a Schwarz method. In all other cases, the matrices* F^{-1} *and* S^{-1} *are approximated, and then a block preconditioner like (3.2) is applied.*

	$\hat{S}^{-1}$	$\hat{F}^{-1}$
PCD	$-\mathbf{A_S}\left[A_p^{-1}\right] F_p \mathbf{A_S}\left[M_p^{-1}\right]$	$\mathbf{A_S}\left[F^{-1}\right]$
SIMPLEC	$\mathbf{A_S}\left[\left(-C - BH_F^{\Sigma}B^T\right)^{-1}\right]$	$\mathbf{A_S}\left[F^{-1}\right]$
LSC	$-\mathbf{A_S}\left[\left(BH_{M_u}^{\Sigma}B^T\right)^{-1}\right]\left(BH_{M_u}^{\Sigma}FH_{M_u}^{\Sigma}B^T\right)\mathbf{A_S}\left[\left(BH_{M_u}^{\Sigma}B^T\right)^{-1}\right]$	$\mathbf{A_S}\left[F^{-1}\right]$
LSC_{A_p}	$-\mathbf{A_S}\left[A_p^{-1}\right]\left(BH_{M_u}^{\Sigma}FH_{M_u}^{\Sigma}B^T\right)\mathbf{A_S}\left[A_p^{-1}\right]$	$\mathbf{A_S}\left[F^{-1}\right]$
$\text{LSC}_{\text{stab},A_p}$	$-\mathbf{A_S}\left[A_p^{-1}\right]\left(BH_{M_u}^{\Sigma}FH_{M_u}^{\Sigma}B^T\right)\mathbf{A_S}\left[A_p^{-1}\right] - \alpha D^{-1}$	$\mathbf{A_S}\left[F^{-1}\right]$
Monolithic	$\mathbf{A_S}\left[\mathcal{F}^{-1}\right]$	

to be approximated for the different preconditioning strategies. Table 5.2 gives an overview of the combinations of the different preconditioning strategies.

6. Numerical results. To examine the performance of the different preconditioners, we consider two three-dimensional problem settings: the well-known backward-facing step and a realistic geometry of an artery from [1]. We show results for stationary and transient simulations with volume force $\mathbf{f} = 0$. In Section 6.2, the monolithic preconditioner is tested for variations of the coarse spaces and further modifications. Subsequently, in Section 6.3, results of block preconditioners are shown, again for different coarse spaces. The preconditioners are always applied as right preconditioners. Finally, in Section 6.4, for specific choices of coarse spaces and modifications, the performance of monolithic and block preconditioners is compared.

The parallel results were obtained on the Fritz supercomputer at Friedrich-Alexander-Universität Erlangen-Nürnberg. Fritz has 992 compute nodes, each with two Intel Xeon Platinum 8360Y *Ice Lake* processors and 256 GB of DDR4 RAM.

For the monolithic as well as the block preconditioners, different coarse spaces can be applied to the velocity and pressure components. The coarse spaces GDSW, GDSW*, and RGDSW (see Section 4.1) are used. This offers the possibility to combine different coarse-space combinations to improve convergence. Similarly to the notation of a discretization, for example, P2-P1, where the first term stands for the velocity and the second for the pressure discretization, we refer to the coarse spaces as, for example, GDSW*-RGDSW. To shorten the notation, sometimes the abbreviations G for GDSW, R for RGDSW, and G* for GDSW* are used in the figures. Different strategies for the reuse of parts of the preconditioner exist; they will be introduced based on the monolithic preconditioner but are equally used for block preconditioners; see Section 6.2.1. Note that we employ a scaled additive Schwarz method in the first level; see Section 4.

Newton's method terminates when the relative residual $\|\mathcal{R}(X^{k+1})\|_2/\|\mathcal{R}(X^0)\|_2$ or the Newton update $\|X^{k+1} - X^k\|_2$ reach a tolerance of 10^{-8}. The initial guess for Newton's method is $X^0 = 0$. For time-dependent problems this refers only to the first time step. Newton's method is used with GMRES along with an adaptive forcing term η_k as described in Section 2.2. The initial forcing term is set to $\eta_0 = 10^{-3}$, the minimum forcing term to $\eta_{\min} = 10^{-8}$, the maximum forcing term to $\eta_{\max} = 10^{-3}$, and the constants to $\alpha = 1.5$ and

ETNA
Kent State University and
Johann Radon Institute (RICAM)

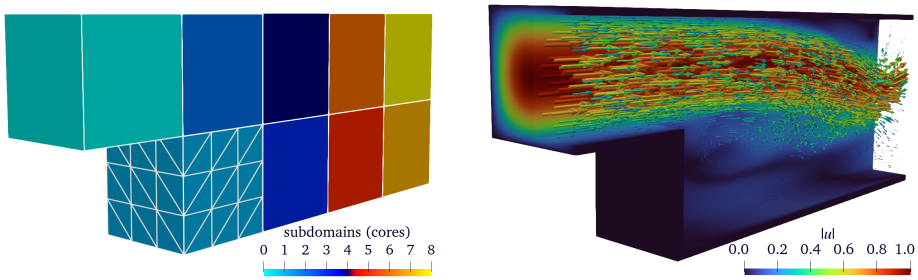

FIG. 6.1. Test case 1: Backward-facing step (BFS) problem. *Left: Backward-facing step geometry with structured mesh partition into 9 subcubes, each with side length* 1 cm. *The subcubes correspond to the subdomains, and in the case displayed, we have* $H/h = 3$. *For a higher number of processor cores, the subcubes are divided up further. For example, for 243 processor cores, each subcube holds* $3 \times 3 \times 3$ *subdomains. Right: Solution to transient BFS problem at* $t = 10.0\,\mathrm{s}$, *using a time-step size* $\Delta t = 0.05\,\mathrm{s}$. *The Reynolds number is* 3 200 *with kinematic viscosity* $\nu = 6.25 \times 10^{-4}\,\mathrm{cm}^2/\mathrm{s}$. *P2-P1 discretization with* $H/h = 9$, *computed on 243 cores.*

$\gamma = 0.9$; cf. (2.6). In each nonlinear iteration, the GMRES iteration is started with zero as the initial guess.

For the Newton method in NOX, we employ a line search with backtracking as the globalization strategy; see [50, Section 5.4] for more details and [23] for a full derivation of the method and globalization techniques. The starting step length is set to 1. The other required parameters are left at their respective default values; see the NOX documentation [79].

The initial mesh partition defining the nonoverlapping subdomains is either structured and constructed manually, or it is constructed by METIS [52], which results in an unstructured partition of the mesh. The subdomains are of similar diameter, which we denote by H; i.e., $H \approx \operatorname{diam}(\Omega_i)$. We choose an algebraically determined overlap of $\hat{\delta} = 1$ for the construction of the local overlapping stiffness matrices; cf. Section 4. For the transient simulations we apply the BDF-2 time stepping scheme; see (2.4) and [37].

Due to comprehensive legends and space restrictions, we occasionally show the legend only in one subfigure if it is valid for all subfigures of the respective figure.

6.1. Test cases. We use two geometries to test the preconditioners: a backward-facing step geometry with a structured mesh that is decomposed into cube-shaped subdomains and a realistic artery with an unstructured mesh and an unstructured domain decomposition.

6.1.1. Test case 1: backward-facing step (BFS). The three-dimensional backward-facing step (BFS) test case is used with a structured mesh and a structured mesh partition; see Figure 6.1 (left) for the partition and Figure 6.1 (right) for the solution of a sample case. We test the weak scaling properties of the monolithic and block preconditioners for stationary and transient flow for this geometry. To this end, the characteristic subdomain size H/h is kept constant, while the number of subdomains increases with the number of processor cores. For the different finite element discretizations and subdomain resolutions, the number of degrees of freedom varies; see Table 6.1. The corresponding coarse-space dimensions are presented in Figure 6.2 for a selection of coarse-space combinations.

We prescribe a parabolic-like inflow profile on the inlet boundary of the domain $\partial\Omega_{\text{in}}$ with a maximum inflow velocity of $v_{\max}$ and an inflow height $L = 1\,\mathrm{cm}$:

$$\mathbf{u}(0, y, z) = \left[\left(16 v_{\max} yz(L-y)(L-z)\right)/L^4, 0, 0\right], \qquad (y, z) \in [0, L]^2.$$

ETNA
Kent State University and
Johann Radon Institute (RICAM)

TABLE 6.1
Subdomain resolution and number of degrees of freedom of the backward-facing step geometry for a structured mesh partition. *Subdomain resolution H/h and number of degrees of freedom for different discretizations and a varying number of subdomains N (processor cores).*

		Number of degrees of freedom			
Discretization	H/h	$N = 243$	$N = 1\,125$	$N = 4\,608$	$N = 15\,552$
Q1-Q1 / P1-P1	17	$4.9 \cdot 10^6$	$22 \cdot 10^6$	$92 \cdot 10^6$	$308 \cdot 10^6$
Q2-Q1 / P2-P1	9	$4.5 \cdot 10^6$	$21 \cdot 10^6$	$85 \cdot 10^6$	$285 \cdot 10^6$
Q2-P1-disc.	9	$5.1 \cdot 10^6$	$23 \cdot 10^6$	$95 \cdot 10^6$	$320 \cdot 10^6$

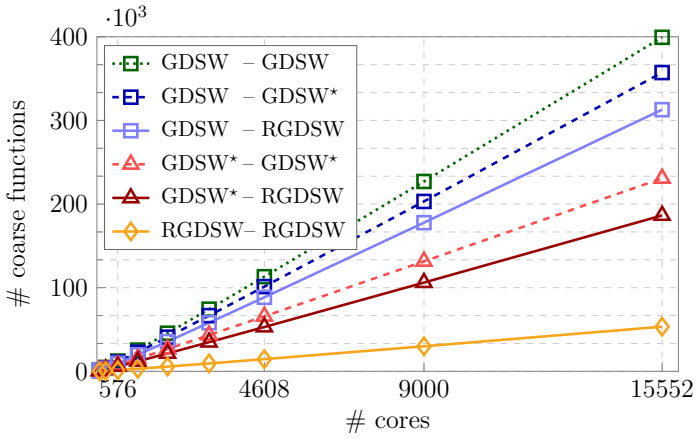

FIG. 6.2. Number of coarse functions for different numbers of processor cores (which equals the number of subdomains) for the backward-facing step (BFS) problem. *The first name corresponds to the interface partition of unity for the velocity component and the second for the pressure component.*

To define the Reynolds number of the BFS problem, we use the outlet height of $2\,\mathrm{cm}$ as the characteristic length and the maximum inflow velocity $v_{\max}$ as v in (3.3):

$$\mathrm{Re} = \frac{2v_{\max}}{\nu}. \tag{6.1}$$

For $v_{\max} = 1\,\mathrm{cm/s}$ and a kinematic viscosity of $\nu = 0.01\,\mathrm{cm}^2/\mathrm{s}$, the Reynolds number is 200. If not stated otherwise, this is the default configuration for the BFS problem.

6.1.2. Test case 2: realistic artery. For the second test case, we consider a tetrahedralized geometry of the realistic artery in Figure 6.3 (left) from [1] with 170 000 P1 nodes and 915 336 elements. The full fluid flow problem with a P2-P1 discretization has 4.1 million degrees of freedom. The focus of this test case is on transient flow simulations where the flow profile contains one heart beat; see Figure 6.3 (right).

The interior (lumen) diameter of the artery varies between $0.2\,\mathrm{cm}$ and $0.34\,\mathrm{cm}$. The full geometry from [1] also contains layers of plaque, media, and adventitia (not shown here). The (outer) diameter of this artery is relatively even and approximately between $0.44\,\mathrm{cm}$ and $0.47\,\mathrm{cm}$. This is in the range of mean diameters of renal arteries as stated in [62, Table 2]; see also [68, Section 3.1], where a catheter with an interior diameter of $0.2\,\mathrm{cm}$ was used to emulate a renal artery. Motivated by this correlation, we prescribe a flow rate and a range for the Reynolds number that is realistic for this kind of artery. The inflow profile at the inlet $\partial\Omega_{\mathrm{in}}$ of the domain is constructed by solving a Poisson problem on the inlet. Since the inlet is

ETNA
Kent State University and
Johann Radon Institute (RICAM)

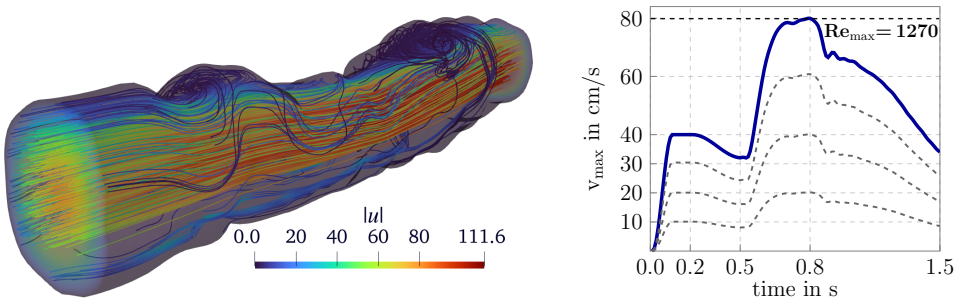

FIG. 6.3. Test case 2: realistic artery problem. *Geometry taken from [1]. Left: Magnitude of the solution* $\mathbf{u}_h$ *at* $t = 0.8\,\mathrm{s}$ *with initial maximum inflow velocity* $v_{\max} = 40\,\mathrm{cm/s}$. *Right: Inflow profile over time of the realistic artery problem. Ramp phase until* $t = 0.1\,\mathrm{s}$ *and subsequent initial maximum inflow velocity* $v_{\max} = 40\,\mathrm{cm/s}$. *The maximum Reynolds number is calculated based on the maximum velocity and on an approximate maximum diameter of artery; see (6.2). The heartbeat and the corresponding flow rate were constructed in [2] based on inflow pressure data from [66].*

TABLE 6.2
Detailed configuration for the realistic artery problem (Test case 2). *Configurations for the maximum initial inflow velocity, the time-step size, and approximate maximum Reynolds number. The maximum Reynolds number for this test is defined as* $\mathrm{Re}_{art,\max}$ *in (6.2).*

$v_{\max}$	$10\,\mathrm{cm/s}$	$20\,\mathrm{cm/s}$	$30\,\mathrm{cm/s}$	$40\,\mathrm{cm/s}$
Δt	$0.005\,s$	$0.0025\,s$	$0.001875\,s$	$0.00125\,s$
$\mathrm{Re}_{\mathrm{art},\max}$	345	665	970	1 270

fairly round, this approximately yields a parabolic flow profile. The proposed blood flow rates in [68, Section 3.3] range between $0.1\,\mathrm{cm}^3/\mathrm{s}$ and $4.0\,\mathrm{cm}^3/\mathrm{s}$ and are constant over time. For the (not perfectly round) inlet of our geometry, these flow rate values translate to a maximum inflow velocity ranging from 3.0 to $120.0\,\mathrm{cm/s}$.

The kinematic viscosity is set to $\nu = 0.03\,\mathrm{cm}^2/\mathrm{s}$. Since the proposed artery has no fixed diameter and the velocity field throughout the artery varies, we estimate the maximum Reynolds number by

$$\mathrm{Re}_{\mathrm{art}}(t) = \frac{D_{\max} \cdot u_{\max}}{\nu}, \qquad \mathrm{Re}_{\mathrm{art},\max} = \max_t \mathrm{Re}_{\mathrm{art}}(t), \tag{6.2}$$

where $D_{\max}$ is the maximum diameter and $u_{\max}$ is the maximum velocity magnitude $|\mathbf{u}|$ attained in a finite element node. Consequently, with the maximum measured diameter of $0.34\,\mathrm{cm}$ and $u_{\max} = 112\,\mathrm{cm/s}$ for a maximum initial inflow velocity of $v_{\max} = 40\,\mathrm{cm/s}$ (after an initial ramp phase until $0.1\,\mathrm{s}$), we have $\mathrm{Re}_{\mathrm{art},\max} = 1\,270$; see Figure 6.3 (left) for a solution plot.

In Table 6.2, based on a selection of initial flow velocities, the corresponding maximum Reynolds numbers and time steps are given. This corresponds to measurements taken from the renal artery of 10 patients in [81, Table 2]. The peak Reynolds number of the patients ranges between 870 and 1 320, with an average value of 1 145.

Using the realistic artery and flow problem, we can further analyze the solvers' abilities to adapt to a large range of Reynolds numbers and to complex flows. Furthermore, strong scaling is tested.

6.2. The monolithic preconditioner and coarse-problem recycling strategies. The monolithic overlapping two-level Schwarz preconditioner is used for the saddle-point prob-

ETNA
Kent State University and
Johann Radon Institute (RICAM)

TABLE 6.3
Number of cores assigned to solve the coarse problem for the results shown in Figure 6.7 for the stationary BFS problem. *Specifically, this corresponds to the P2-P1 discretization with the GDSW*-RGDSW coarse-space combination. For RGDSW-RGDSW fewer cores are used due to the lower number of coarse functions; compare with Figure 6.2.*

# cores	243	576	1 125	1 944	3 087	4 608	9 000	15 552
# cores for coarse solve	4	4	6	6	8	12	16	16

TABLE 6.4
Comparison of setup and solve times of the monolithic preconditioner if recomputing it the first k Newton iterations. *Reuse of symbolic factorization (SF) and coarse basis (CB). Recompute preconditioner (except for (SF) and (CB)) up to k Newton iterations, reuse entire preconditioner based on the kth iteration afterwards. Stationary BFS problem with P2-P1 discretization with $H/h = 9$. Average GMRES iteration count per Newton step. Total number of Newton iterations required to reach convergence is 5. RGDSW-RGDSW coarse-space combination for the velocity (first component) and pressure (second component).*

	k	5	4	3	2
RGDSW-RGDSW	# avg. iter.	52.8	52.8	54.0	78.4
	Setup cost	29.4 s	24.3 s	19.4 s	14.8 s
	Solve cost	35.9 s	34.9 s	35.9 s	56.1 s
	Total	65.3 s	59.2 s	**55.3** s	70.9 s

lem (2.5) as described in Section 4.3. In [41] the GDSW and RGDSW coarse spaces were evaluated and compared for three-dimensional Navier–Stokes problems. We will take into account the authors' findings about strategies for reusing parts of the preconditioner (cf. [41, Section 5.4]). In particular, we employ the strategy of reusing the coarse basis functions, which produced the lowest total time—consisting of solve and setup time—in [41]. The problems of the first level and the coarse problem of the Schwarz preconditioner are solved sequentially; for example, the N subdomain problems of the first level are solved on N processor cores, and subsequently a subset of these cores are used to solve the coarse problem in parallel. We do not employ a hybrid Schwarz approach but use a fully additive one; see Section 4. Further improvements with respect to the computation time can be achieved by using additional cores to solve the coarse problem; cf. [41, Section 4.4]. The number of processor cores assigned to the coarse solve depends on the coarse-space dimension (shown in Figure 6.2 for the backward-facing step problem) and ranges between 2 and 16; see Table 6.3 for details in case of the BFS problem.

6.2.1. Recycling strategies. Figure 6.2 displays the dependence of the number of coarse functions on the used coarse spaces. Generally, the higher the number of coarse functions is, the higher the setup costs are. The reuse strategies described in [41] and included in the software package `FROSch` are the reuse of the symbolic factorization of $\mathcal{F}_0$ (SF), of the coarse basis functions ϕ (CB), and of the coarse matrix $\mathcal{F}_0$ (CM).

The more extensive the reuse strategy is, the more it affects the linear iteration count. The reuse of the symbolic factorization (SF) is always applied. In general, the preconditioner is rebuilt or updated in each Newton iteration. If the coarse basis is reused (CB), then only the coarse matrix is recomputed in each Newton iteration.

Additionally to the previously described reuse strategies, the setup cost can be further reduced by only rebuilding or updating the preconditioner in the first k Newton iterations (of each time step or of the stationary simulation) and then reusing it. This is crucial for the monolithic preconditioner, where the setup cost is higher compared to block-preconditioning strategies.

ETNA
Kent State University and
Johann Radon Institute (RICAM)

Based on the results in Table 6.4 for the stationary BFS case, we can save computing time by reusing the complete preconditioner after three Newton steps. For $k = 3$ we achieve the best total time since the linear iteration count is only slightly increased while the setup time is small. The reuse of the coarse basis is not always the best choice, for example, for the GDSW-GDSW coarse-space combination, since it increases the solve time to the point where it outweighs the time savings in the setup part. As it is preferable to use the RGDSW coarse space due to its smaller dimension, for the stationary tests we employ the CB reuse strategy and reuse the preconditioner in the Newton method after iteration $k = 3$. In transient simulations it is sufficient to update the preconditioner only in the first ($k = 1$) Newton step of each time step, probably since the required average number of Newton iterations per time step is only approximately three. Furthermore, the CB reuse strategy is applied in transient simulations over all time steps; that is, the coarse basis is only computed once for the entire simulation, but due to the changing system matrix, the coarse matrix, and thus the preconditioner, is updated once in each time step. A justification for the reuse of the coarse basis is also given later in Figure 6.13, which displays a direct comparison of reusing and not reusing the coarse basis for building the preconditioner. It reveals only a slight difference with respect to the average iteration count.

The stated recycling strategies are also applied to the block preconditioners. Additionally, for the LSC and PCD block preconditioners, we reuse the approximations of A_p^{-1} and M_p^{-1} that are used for the construction of the Schur complement approximation since they are independent of the solution and only need to be built once.

Regarding the construction of the coarse matrix for the monolithic two-level overlapping Schwarz preconditioner, we can additionally choose between using the unaltered coarse basis (4.6) (*full* ϕ) or excluding the off-diagonal blocks:

$$\phi \leftarrow \begin{bmatrix} \phi_{\mathbf{u},\mathbf{u_0}} & 0 \\ 0 & \phi_{p,p_0} \end{bmatrix}.$$

This can have an effect on the scaling behavior of the preconditioner and will be further discussed in the following sections.

6.2.2. The stationary Navier–Stokes problem. Different finite element discretizations are considered for the stationary BFS problem. First, we consider the P1-P1 discretization with the Bochev–Dohrmann stabilization and the inf-sup stable P2-P1 discretization for a comparison of different coarse-space combinations and to show results of the new GDSW* coarse space. For the inf-sup stable discretizations, formulation (4.7) with the pressure projection is used. For results omitting the pressure projection in case of stable discretizations, see Figure A.2 in the appendix.

In Figure 6.4 we observe that P1-P1 elements yield similar results for all coarse-space combinations, independent of using the full ϕ or excluding the off-diagonal blocks to construct the coarse matrix. Weak scaling can be observed for up to 15 552 cores; see also Figures 6.6 and 6.7 for results of RGDSW-RGDSW up to 15 552 cores. For the stable P2-P1 element, in contrast, we observe a substantial influence in Figure 6.5 for some combinations of coarse spaces and using the full or decoupled ϕ.

We first focus on the results in Figure 6.5 (left), where the off-diagonal blocks of ϕ are excluded. It stands out that, if RGDSW for the pressure is combined with a larger coarse space like GDSW or GDSW* for the velocity, then we achieve the lowest average GMRES iteration count. If a smaller coarse space for the velocity is used than for the pressure, then the algorithm does not reach the GMRES convergence criterion (results are not shown here). The comparison to an inf-sup condition of a finite element discretization seems natural. On the other hand, the "equal-order" approach RGDSW-RGDSW is scalable with a low number of

ETNA
Kent State University and
Johann Radon Institute (RICAM)

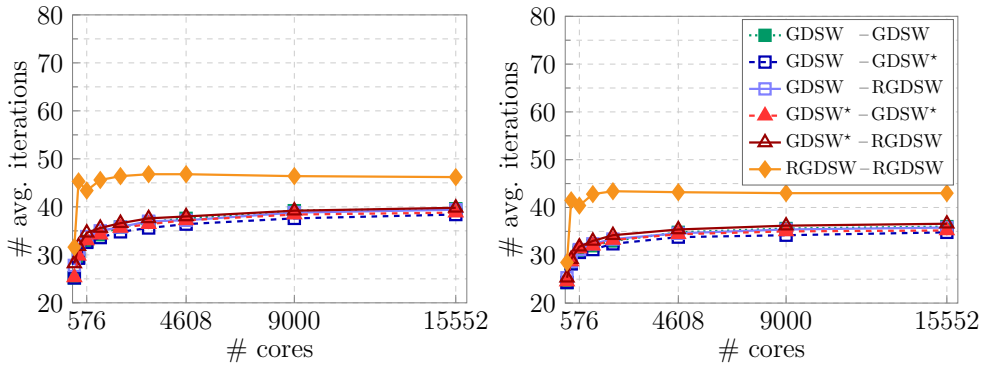

FIG. 6.4. Comparison of different coarse-space combinations for the velocity (first component) and pressure (second component) for a P1-P1 discretization and the monolithic preconditioner. *Average GMRES iteration count per Newton step. Total number of Newton iterations required to reach convergence is 5. Weak scaling test. Stationary BFS problem with $H/h = 17$. $Re = 200$. Pressure projection is not used. Left: Excluding off-diagonal blocks in ϕ to build the coarse matrix. Right: Using full ϕ to build the coarse matrix.*

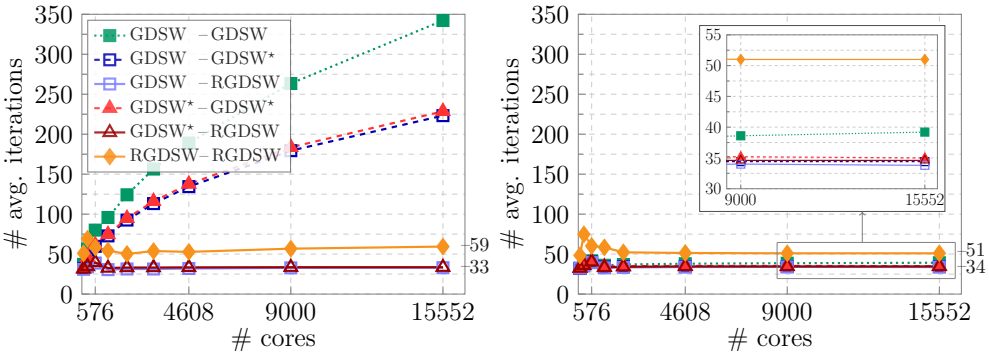

FIG. 6.5. Comparison of different coarse-space combinations for the velocity (first component) and pressure (second component) for a P2-P1 discretization and the monolithic preconditioner. *Average GMRES iteration count per Newton step. Total number of Newton iterations required to reach convergence is 5. Weak scaling test. Stationary BFS problem with $H/h = 9$. $Re = 200$. Pressure projection included in the first level; see Figure A.2 in the appendix for results if the pressure projection is not used. See Figure A.1 in the appendix for results of a lid-driven cavity problem. Left: Excluding off-diagonal blocks in ϕ to build the coarse matrix. Right: Using full ϕ to build the coarse matrix.*

iterations, albeit not the lowest one. The cause for the correlation of the iteration count and the relative size of the velocity and pressure coarse spaces may also be independent of an inf-sup condition; further research is required.

For the results using the full ϕ in Figure 6.5 (right), we observe similar good scaling results but for all coarse-space combinations. If the full ϕ is used to construct the coarse matrix, then it seems imperative that the pressure projection is also applied in the first level; if the pressure projection is omitted, then we lose scalability for all combinations in Figure A.2 (right). On the other hand, the analogue to Figure 6.5 (left) is Figure A.2 (left), which shows the same general scaling behavior; that is, the pressure projection does not seem to affect the case in which ϕ is decoupled. Independent of the use of the decoupled or full ϕ, GDSW*-RGDSW is the best-performing configuration for the P2-P1 finite element discretization if the pressure projection is used. It remains to be thoroughly investigated, however, to determine how the various settings and coarse spaces impact each other. Figure A.1 in Appendix A qualitatively shows the same results as Figure 6.5 for a lid-driven cavity, indicating that the results are not

ETNA
Kent State University and
Johann Radon Institute (RICAM)

TABLE 6.5
Comparison of setup and solve time of the three best coarse-space combinations for P1-P1 and P2-P1 discretization on 4 608 cores. *Stationary BFS problem with configurations according to Table 6.1. Total number of Newton iterations required to reach convergence is 5. Use of Bochev–Dohrmann stabilization for P1-P1 elements. Use of full ϕ. Use of pressure projection for P2-P1. Re* $= 200$.

		GDSW-RGDSW	GDSW*-RGDSW	RGDSW-RGDSW
	# avg. iterations	34.6	35.4	43.2
P1-P1	Setup time	26.7 s	17.5 s	12.8 s
	Solve time	21.3 s	18.4 s	22.3 s
	Total	48.0 s	35.9 s	**35.1** s
	# avg. iterations	34.4	34.6	51.4
P2-P1	Setup time	47.6 s	23.7 s	20.2 s
	Solve time	27.3 s	22.5 s	32.8 s
	Total	74.9 s	**46.2** s	53.0 s

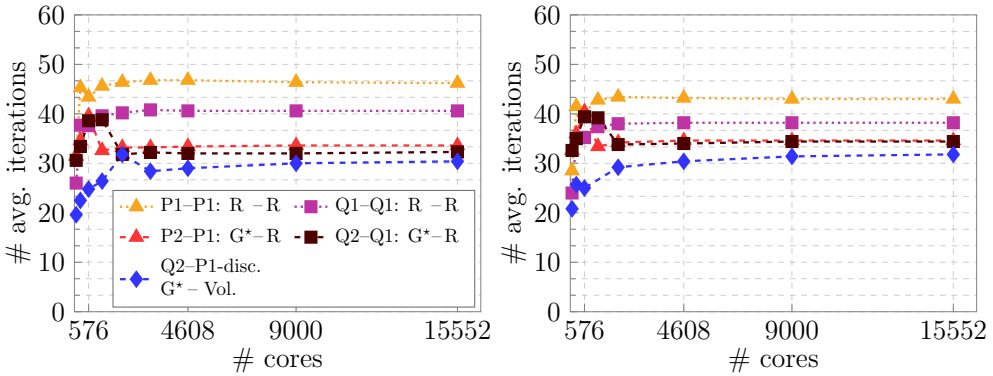

FIG. 6.6. Comparison of different finite element discretizations for the monolithic preconditioner. *Combinations of different coarse spaces for the velocity (first component) and pressure (second component). G: GDSW. G*: GDSW*. R: RGDSW. Average GMRES iteration count per Newton step. Total number of Newton iterations required to reach convergence is 5. Weak scaling test. Stationary BFS problem with configurations according to Table 6.1. Re* $= 200$. *Use of Bochev–Dohrmann stabilization for P1-P1 and Q1-Q1 elements. Pressure projection included in the first level for P2-P1 and Q2-Q1 discretization. Left: Excluding off-diagonal blocks in ϕ to build the coarse matrix. Right: Using full ϕ to build the coarse matrix.*

specific to the backward-facing step problem.

Table 6.5 provides details corresponding to Figure 6.4 (right) (P1-P1 discretization) and Figure 6.5 (right) (P2-P1 discretization) for a selection of coarse-space combinations. Even though the average iteration count for the RGDSW-RGDSW combination is highest for P1-P1, it offers the lowest total cost due to its low setup cost. Notably, the setup cost for GDSW is higher than for GDSW*. The lowest setup cost offers the RGDSW coarse space. In Table 6.5 we observe that GDSW*-RGDSW performs best with respect to the total time (setup+solve) for the P2-P1 discretization.

For a comparison of all the different discretizations, P1-P1, Q1-Q1, P2-P1, Q2-Q1, and Q2-P1-disc., we focus on GDSW*-RGDSW and RGDSW-RGDSW (see Figure 6.6), based on the previous findings of which combination gave the best results.

Due to the discontinuous pressure in Q2-P1-disc., there is no interface for the pressure component. As a result, all coarse spaces define the same space on the pressure: the span

ETNA
Kent State University and
Johann Radon Institute (RICAM)

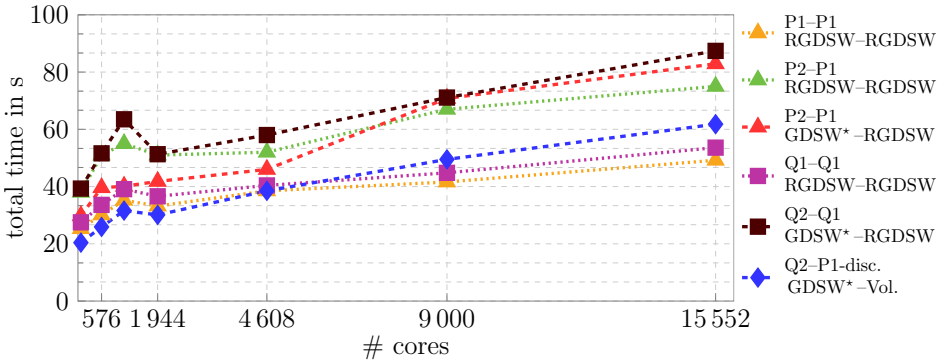

FIG. 6.7. Comparison of total time (setup + solve time) for different finite element discretizations for the monolithic preconditioner. *Combinations of different coarse spaces for the velocity (first component) and pressure (second component). Average GMRES iteration count per Newton step. Total number of Newton iterations required to reach convergence is 5. Weak scaling test; see Table 6.3 for the number of cores used to solve the coarse problem based on GDSW⋆-RGDSW. Stationary BFS problem with configurations according to Table 6.1. Re = 200. Use of Bochev–Dohrmann stabilization for P1-P1 and Q1-Q1 elements. Pressure projection included in the first level for P2-P1 and Q2-Q1 discretization. Use of full ϕ.*

of constant functions on each subdomain; see Section 4.1.3. As mentioned at the beginning of this section, if the pressure coarse space is too large in comparison to the velocity coarse space, then the algorithm does not converge. We observe the same for Q2-P1-disc. if RGDSW is used for the velocity. The explanation may be the same as before since, for example, in the case of 243 subdomains, the RGDSW pressure coarse space for a continuous P1 pressure space has 124 coarse functions (vertices of the domain decomposition), but any GDSW-type coarse space for P1-discontinuous pressure functions has 243 coarse functions. Thus, RGDSW-RGDSW for Q2-P1 has a smaller pressure coarse space than for Q2-P1-disc. Consequently, the Q2-P1-disc. test results are based on the GDSW⋆ coarse space for the velocity.

The different configurations all yield similar good weak scaling results for up to 15 552 processor cores. The stabilized P1-P1 and Q1-Q1 discretizations produce slightly higher average GMRES iteration counts than their stable counterparts. Excluding the off-diagonal blocks in ϕ or using it fully has only a minor impact on the results (compare Figure 6.6 (left) and (right)). Corresponding to Figure 6.6 (left), Figure 6.7 displays the total time, consisting of setup and solve time. Again, all the different configurations yield similar good weak scaling results with respect to the total time for up to 15 552 processor cores. Included are timings for P2-P1 and RGDSW-RGDSW, a combination that was not included in Figure 6.6 but displayed previously in Figure 6.5; the results show that for a smaller number of subdomains, the number of iterations and the total time using GDSW⋆ for the velocity is smaller than if RGDSW is used. However, the timings show that this changes for a larger number of subdomains.

6.2.3. The transient Navier–Stokes problem. For the transient backward-facing step problem, we simulate up to $t = 10.0\,\mathrm{s}$. We choose a time-step size of $\Delta t = 0.05\,\mathrm{s}$, giving 200 time steps in total (after which an almost stationary solution is reached). We omit the pressure projection in the transient case, as it only slightly influences linear convergence but increases the computation time. We use the coarse-space combinations that performed best in the stationary case. Furthermore, we focus on the construction of the coarse matrix where the off-diagonal blocks of ϕ are omitted.

In Figure 6.8 we investigate how the average iteration count of the linear solver (over all Newton and time steps) depends on the Reynolds number; see Table B.1 for details. See (6.1) for the Reynolds number definition of the BFS problem. The Reynolds number

ETNA
Kent State University and
Johann Radon Institute (RICAM)

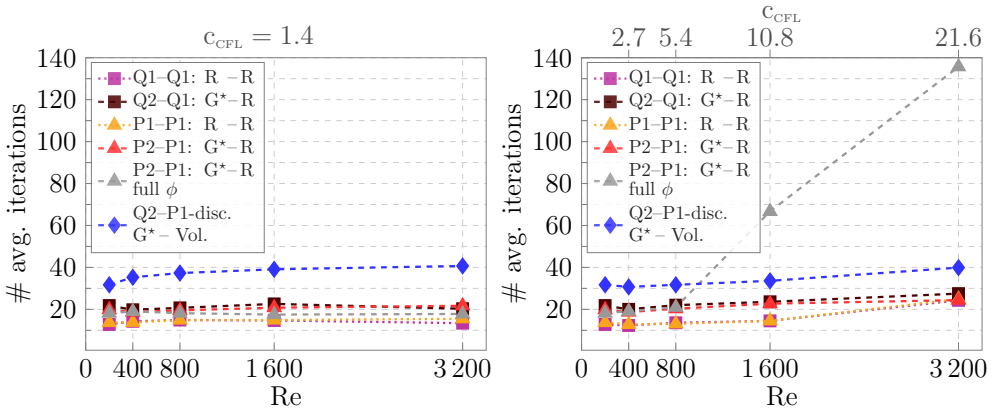

FIG. 6.8. Comparison of different finite element discretizations for the monolithic preconditioner for an increasing Reynolds number. *For the definition of the Reynolds number, see* (6.1). *Average GMRES iteration count per Newton step. Transient BFS problem with configurations according to Table 6.1. Time step size is constant,* $\Delta t = 0.05\,\mathrm{s}$, *simulation until* $t = 10\,\mathrm{s}$, *which amounts to 200 time steps in total. Use of Bochev–Dohrmann stabilization for P1-P1 and Q1-Q1 elements. 243 processor cores. Combinations of different coarse spaces for the velocity (first component) and pressure (second component). G: GDSW. G*: GDSW*. R: RGDSW. No use of pressure projection. Exclude off-diagonal blocks in* ϕ *except for one case (see legend). For detailed results, see Table B.1. Left: Increasing Reynolds number with constant* $v_{\max} = 1\,\mathrm{cm/s}$. *The viscosity* ν *is decreased from* $0.01\,\mathrm{cm^2/s}$ *for* $Re = 200$ *to* $\nu = 6.25 \times 10^{-4}\,\mathrm{cm^2/s}$ *for* $Re = 3\,200$. *The CFL number is* 1.4. *Average number of Newton iterations is between 3.0 and 3.1. Right: Increasing Reynolds number with constant* $\nu = 0.01\,\mathrm{cm^2/s}$. *The velocity* $v_{\max}$ *is increased from* $1\,\mathrm{cm/s}$ *for* $Re = 200$ *to* $16\,\mathrm{cm/s}$ *for* $Re = 3\,200$. *The CFL number increases with higher velocity. Average number of Newton iterations is between 3.0 and 4.1.*

can be increased in two ways: either by decreasing the viscosity ν (see Figure 6.8 (left)) or by increasing the maximum velocity (see Figure 6.8 (right)). The solver and preconditioner combination is very robust with respect to the Reynolds number.

Both approaches of adjusting the Reynolds number yield similar results for the monolithic preconditioner if the off-diagonal blocks of ϕ are omitted; this is not the case if the full ϕ is used or for the block preconditioners PCD and SIMPLEC, as we will see in Section 6.4. When the CFL number increases (Figure 6.8 (right)), the approach using the full ϕ is not robust. However, rebuilding the coarse basis instead of reusing it will improve these results such that the number of iterations is comparable to the other coarse-space variants. Nonetheless, since reusing the coarse basis is computationally more efficient and omitting the off-diagonal blocks proved to be more robust, we continue to use that strategy. It may be, however, that in different circumstances rebuilding the coarse basis ϕ in each Newton step and not discarding the off-diagonal blocks is advantageous; this needs to be further investigated.

The insight that the CFL number is the relevant metric in the considered case of varying the Reynolds number in different ways is also reflected by considering the dimensionless form of the Navier–Stokes equations. There, $\mathbf{u}$ is transformed to $\mathbf{u}^* = \mathbf{u}/u_{\mathrm{ref}}$ via a reference velocity (for example, the maximum velocity at the inlet). The relevant influence on the solver by means of changing the Reynolds number via the velocity or the viscosity is then found in the transformed time step $\Delta s = u_{\mathrm{ref}} \Delta t / L$, where L is a characteristic length scale (see [29, p. 334 and p. 412] for details). This is comparable to a CFL number of the type $u_{\mathrm{ref}} \Delta t / \Delta x$ since L and Δx are fixed.

We remark that using a high CFL number may be inadvisable, even for stable time discretizations, depending on the desired accuracy of the numerical solution. It is, however, not a focus of this work to address this but to analyze the robustness of the solvers with respect to different parameters.

ETNA
Kent State University and
Johann Radon Institute (RICAM)

TABLE 6.6
Comparison of block preconditioners for different Reynolds numbers for the stationary BFS problem. *P2-P1 discretization with $H/h = 9$, 1 125 cores. (BC-2) is applied in PCD and LSC$_{A_p}$. RGDSW-RGDSW coarse spaces for the velocity (first component) and pressure (second component). Average GMRES iteration count per Newton step and number of Newton steps in parentheses.*

Preconditioner	Diagonal	BFBt	LSC	LSC$_{A_p}$	PCD	SIMPLEC
Re = 20	127 (4)	180 (4)	169 (4)	154 (4)	82 (4)	278 (4)
Re = 200	266 (5)	292 (5)	257 (5)	207 (5)	103 (5)	316 (5)

TABLE 6.7
Comparison of boundary conditions in the PCD block preconditioner for the stationary BFS problem. *For different boundary conditions applicable to PCD, see Table 3.2. P2-P1 discretization with $H/h = 9$, 243 cores, and Re = 200. RGDSW-RGDSW coarse-space combination for the velocity and Schur complement components. Average GMRES iteration count per Newton step.*

Boundary condition strategy	(BC-1)	(BC-2)	(BC-3)
# avg. iterations	> 500	87.7	93.7

6.3. Block preconditioners. We conduct similar tests for the block preconditioners as for the monolithic preconditioner. A description of the used recycling strategies for block preconditioners is given in Section 6.2.1. First, the stationary BFS problem is used to analyze the different block preconditioners with respect to weak scaling. Then, weak scaling and the robustness with respect to an increasing Reynolds number is analyzed for the transient BFS problem.

In the stationary case, we approximate the inverse of M_p arising in the PCD preconditioner with the diagonal approximation that is denoted *AbsRowSum*; see Section 5.3.1. There it was advantageous to approximate the inverse of M_p with a Schwarz method. For the transient case though, approximating M_p with a two-level overlapping Schwarz method proved to be best. The inverse of M_u arising in the LSC-type preconditioners is also approximated via the *AbsRowSum* diagonal scaling; see Section 5.3.2.

6.3.1. The stationary Navier–Stokes problem. Similar to Section 6.2.2 we use the Test case 1 with the backward-facing step in three dimensions. First we compare in Table 6.6 the available block preconditioners from `Teko` and a diagonal block preconditioner implemented in the `FEDDLib`. The diagonal block preconditioner approximates the Schur complement with $-\frac{1}{\nu}M_p$; the inverse of the approximate Schur complement and the inverse of F^{-1} are then approximated with the Schwarz method. The BFBt method is defined by (3.14), as it is a variant of the LSC block preconditioner.

The SIMPLE preconditioner is known to not perform well in the stationary case; see, for example, [12, Section 4.2], [25, Section 3], or [74, Section 8.3]. Only the PCD block preconditioner delivers moderate results. The results for LSC may improve by using an additional scaling, like the ones introduced in [9, 31]. However, we did not see an improvement with the scaling in [9] for our setting, and an implementation of the scaling matrix from [31] is not available to us—see [31] for a derivation of a scaling matrix in a two-dimensional setting. Consequently, and particularly later when we compare monolithic with block preconditioners, we focus mainly on the PCD preconditioner and at times show results based on the SIMPLEC preconditioner for reference.

In Section 4.2 we have shown how the two-level overlapping Schwarz preconditioners are applied to block systems. The arising inverses $\hat{F}^{-1}$ and $\hat{S}^{-1}$ of the block preconditioners in Section 3 are approximated with different Schwarz preconditioners.

ETNA
Kent State University and
Johann Radon Institute (RICAM)

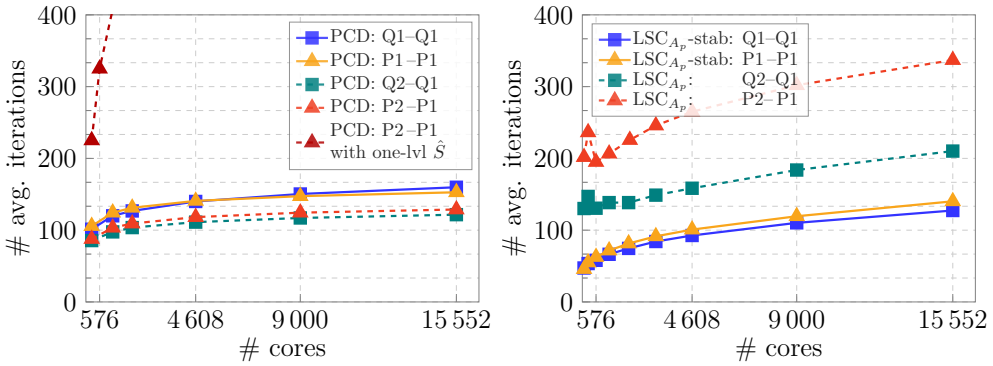

FIG. 6.9. Comparison of different finite element discretizations for block-triangular preconditioners. *Use of RGDSW-RGDSW (R-R) coarse space for the velocity (first component) and for Schur complement components (second component) for both PCD and LSC (except for one result using PCD; see legend). (BC-2) boundary condition strategy used for PCD and LSC. Average GMRES iteration count per Newton step. Total number of Newton iterations required to reach convergence is 5. Weak scaling test. Stationary BFS problem with configurations according to Table 6.1. $Re = 200$. Use of Bochev–Dohrmann stabilization for P1-P1 and Q1-Q1 elements. No use of pressure projection. Left: PCD block preconditioner. One-level Schwarz approximation for $\hat{S}^{-1}$ denoted as one-lvl $\hat{S}$. Right: LSC$_{A_p}$ block preconditioner; see Section 3.2.*

For the stationary case, using the boundary condition strategy (BC-2) for PCD proved to be slightly better than the default (BC-3); see Table 6.7. Consequently (BC-2) will be used for the PCD preconditioner for the stationary BFS problem, but we will revert to the default (BC-3) for transient simulations.

We select the RGDSW-RGDSW coarse-space combination as the default strategy: Table A.1 in the appendix shows that different coarse-space combinations deliver similar results if the Reynolds number is varied by changing the viscosity. Furthermore, the setup and solve times are comparable; mixing different coarse spaces for PCD, for example GDSW*-RGDSW, does not improve the average iteration count as in the monolithic case. Note that RGDSW has the lowest setup cost since it has the smallest coarse-space dimension. Especially for a higher number of processor cores, this will become visible; cf. Figure 6.11 (right).

Due to the decoupling of the subproblems via the block-preconditioning approach, the choice of the coarse space for the two-level overlapping Schwarz method is then secondary as indicated by the results in Figure 6.10 and Table A.1. These results also show that the PCD block preconditioner is robust once the boundary conditions are correctly specified (cf. Section 3.1.2).

In Figure 6.9 weak scaling results up to 15 552 processor cores are displayed for the block-triangular preconditioners PCD and LSC. Most results are based on the application of a two-level Schwarz method to approximate $\hat{F}^{-1}$. For reference, we also show a combination that uses a one-level Schwarz approximation for $\hat{S}^{-1}$. We test the inf-sup stable P2-P1 and Q2-Q1 and the stabilized P1-P1 and Q1-Q1 elements but omit the Q2-P1-disc. element since the implementation of the discrete pressure Laplace operator A_p cannot be used for discontinuous elements. In that case, a different type of operator is required for PCD and LSC$_{A_p}$; its implementation is not available to us, and its definition is not straightforward; see [29, pp. 368–370].

For PCD we observe weak scalability for all discretizations in Figure 6.9 (left). Using only a one-level overlapping Schwarz preconditioner for the Schur complement proves to be inadequate; the average iteration count exceeds 400.

ETNA
Kent State University and
Johann Radon Institute (RICAM)

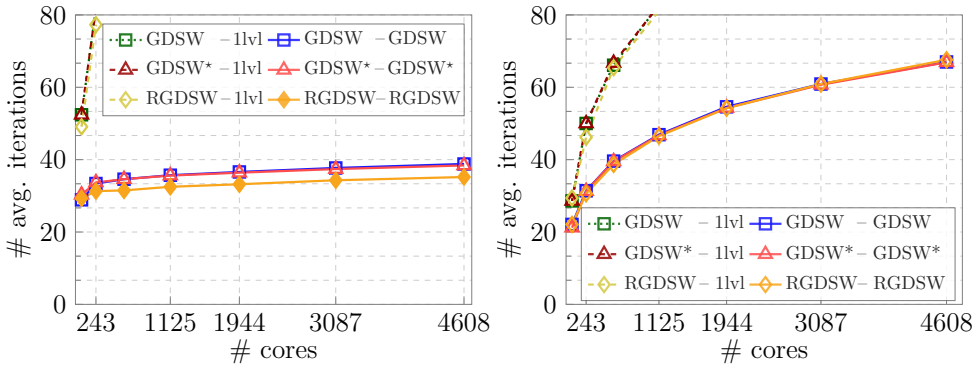

FIG. 6.10. Comparison of different coarse-space combinations for the velocity (first component) and Schur complement components (second component) for a P2-P1 discretization for block preconditioners. *Weak scaling test for different combinations of one- and two-level approaches. Average GMRES iteration count per Newton step over all time steps. Average number of Newton iterations per time step required to reach convergence is 3. Transient BFS problem with* $H/h = 9$. *Time step* $\Delta t = 0.02\,\mathrm{s}$, *simulation until* $t = 1.0\,\mathrm{s}$, *which amounts to 50 time steps.* $Re = 200$. *The CFL number varies due to the increasing mesh resolution and constant time-step size. The maximum CFL number is* 1.44 *for 4 608 cores. Left: Pressure convection-diffusion (PCD) block preconditioner with (BC-3) boundary condition strategy for the Schur complement setup. Right: SIMPLEC block preconditioner.*

The results using LSC are less clear; see Figure 6.9 (right) and Figure A.3. $\mathrm{LSC}_{\mathrm{stab},A_p}$, $\mathrm{LSC}_{\mathrm{stab}}$, and PCD show comparably good scaling results for the stabilized Q1-Q1 and P1-P1 elements; see Figures 6.9 and A.3. However, unstabilized LSC and LSC_{A_p} for stable discretizations do not scale well; for a P2-P1 discretization, both exceed 300 average GMRES iterations. Using Q2-Q1 elements, the convergence for LSC_{A_p} is significantly better than for P2-P1 but still significantly worse than for equal-order elements. Both discretizations show the h-sensitivity of the LSC preconditioner; compare, for example, with [9, Table 4]. Only the stabilized LSC variants are close to reaching an asymptotically constant number of GMRES iterations. See also [74] (e.g., Table 11) for some related results of the LSC preconditioner and the stationary BFS problem or [9] for a comparison of PCD and LSC preconditioners for a lid-driven cavity problem using Q2-Q1 elements.

6.3.2. The transient Navier–Stokes problem. As the PCD preconditioner gave consistently good results in the previous section, we focus our analysis on it for the transient backward-facing step problem but also show results for SIMPLEC as a reference. We iterate for 50 time steps with the step size $\Delta t = 0.02\,\mathrm{s}$.

As stated in Section 3.1.2, different boundary conditions applied to the Schur complement components in the PCD preconditioner can affect the performance. Unlike for the stationary case in Table 6.7, where strategy (BC-2) worked best and (BC-1) failed to converge in a decent number of iterations, for the transient case, all three strategies perform equally well; see Table B.2 in the appendix. Thus, we revert to the default setting of (BC-3) for transient simulations.

We analyze the weak scaling properties of the PCD and SIMPLEC block preconditioners with Figure 6.10. As for the stationary problem, using a one-level method for the pressure component is infeasible due to very high iteration counts. The other shown variants of PCD and SIMPLEC qualitatively perform similarly as in the stationary case; PCD is weakly scalable and does significantly better than SIMPLEC.

6.4. Comparison of specific monolithic and block preconditioners. For the comparison of monolithic and block preconditioners, we only address the configurations that gave the most robust results in the previous sections. Furthermore, we focus mostly on PCD as a block

ETNA
Kent State University and
Johann Radon Institute (RICAM)

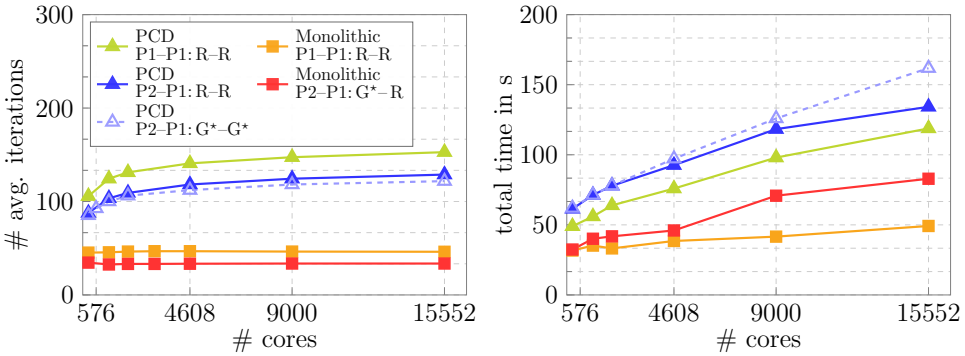

FIG. 6.11. Weak scaling comparison of block and monolithic preconditioners. *Use of GDSW⋆ (G⋆) and RGDSW (R) coarse spaces for the velocity (first component) and pressure (second component). Stationary BFS problem with configurations according to Table 6.1. Re = 200. Use of Bochev–Dohrmann stabilization for P1-P1 elements. Pressure projection included in the first level for monolithic preconditioner with P2-P1 discretization. Exclude off-diagonal blocks in ϕ for monolithic preconditioner. Boundary condition strategy for PCD is (BC-3). Left: Average GMRES iteration count per Newton step. Total number of Newton iterations required to reach convergence is 5. Right: Total time in seconds consisting of setup and solve time.*

preconditioner but show a number of SIMPLEC results for reference. We contrast the weak scaling behavior for the backward-facing step and the strong scaling behavior for the realistic artery. We also study the preconditioners' robustness by increasing the Reynolds number. This is especially relevant for physically realistic scenarios. In [14] similar aspects relevant to hemodynamics simulations were analyzed; among other tests, the authors compared the SIMPLE and PCD block preconditioners for varying viscosities and Reynolds numbers.

6.4.1. Test case 1: backward-facing step.

Stationary simulation. In Figure 6.11 the average iteration count and the total time for the PCD and monolithic preconditioner are displayed. Since the average iteration count of the monolithic preconditioner is significantly lower, the total time is much lower as well. Despite a slightly lower number of iterations, the RGDSW-RGDSW combination for PCD performs better in case of the P2-P1 discretization and a large number of processor cores compared to the GDSW⋆-GDSW⋆coarse-space combination, which can be attributed to the smaller cost associated with the coarse problem.

It stands out that stabilized P1-P1 elements lead to smaller computation times compared to P2-P1, despite a larger iteration count and a slightly larger number of degrees of freedom (cf. Table 6.1). This is probably due to a more favorable sparsity pattern. However, since a P2-P1 discretization should give a better finite element approximation, one cannot infer from the results that P1-P1 elements are preferable.

Transient simulation. The transient backward-facing step simulation consists of 200 time steps with $\Delta t = 0.05\,\mathrm{s}$ until $t = 10\,\mathrm{s}$ is reached. We restrict ourselves to a P2-P1 discretization and the monolithic preconditioner with the GDSW⋆-RGDSW or RGDSW-RGDSW coarse-space combination; PCD and SIMPLEC always use RGDSW-RGDSW. For the comparison of monolithic and block preconditioners, we consider three aspects: firstly, we compare the robustness with respect to an increasing Reynolds number. The Reynolds number is varied by adjusting the viscosity. As we have seen before, changing the velocity instead is more challenging, as this also changes the CFL number. In Figure 6.12 (left), the average linear iteration count is displayed for an increasing Reynolds number. Detailed iteration counts and timings are included in the appendix in Table C.1. The PCD and monolithic preconditioners are robust with respect to the increase of the Reynolds number, but also SIMPLEC shows

ETNA
Kent State University and
Johann Radon Institute (RICAM)

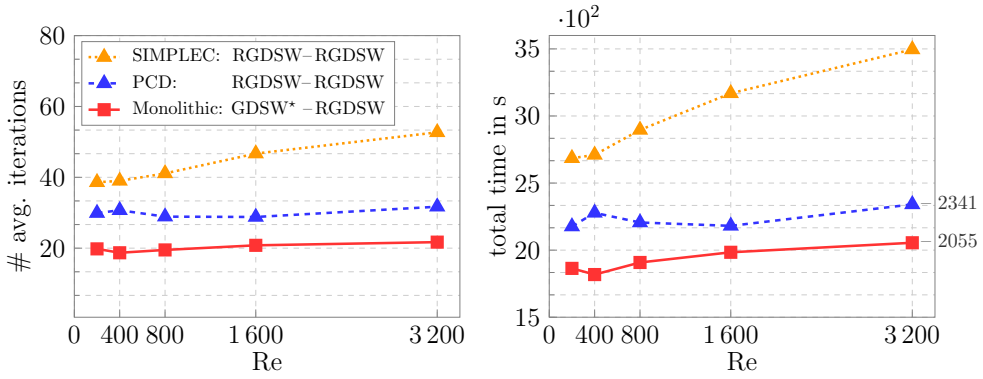

FIG. 6.12. Comparison of block and monolithic preconditioners for an increasing Reynolds number. *Use of GDSW⋆ and RGDSW coarse spaces for the velocity (first component) and pressure (second component). Transient BFS problem with P2-P1 discretization and $H/h = 9$. Increasing Reynolds number due to decreasing viscosity. No pressure projection is used. Exclude off-diagonal blocks in ϕ for monolithic preconditioner. Computation of 200 time steps with time-step size $\Delta t = 0.05\,\mathrm{s}$, simulation until $t = 10\,\mathrm{s}$. CFL number is constant and equal to 1.35. Computations on 243 cores. Boundary condition strategy in PCD is (BC-3). See Table C.1 for detailed numbers corresponding to the figure. Left: Average GMRES iteration count per Newton step per time step. Average number of Newton iterations per time step required to reach convergence is 3. Right: Total time in seconds consisting of setup and solve time.*

only a small increase in the number of iterations. Despite using a larger coarse space for the velocity component, the monolithic preconditioner requires the least amount of CPU time; see Table C.1 and Figure 6.12 (right).

Secondly, we consider robustness with respect to an increasing CFL number. In Figure 6.12, the CFL number remained constant. In Figure 6.13, we raise the Reynolds number by increasing the maximum inflow velocity, which increases the CFL number since the time-step size remains constant. For the monolithic preconditioner, the average iteration count per Newton step over all time steps is still almost constant over the range 1.35 to 21.6 of CFL numbers. PCD and SIMPLEC, however, show a large increase of the average iteration count with the increasing CFL number. A sensitivity of PCD and SIMPLEC with respect to the CFL number was also observed in [12, Figure 9]. Not reusing the coarse basis, as stated in Section 6.2.1, has almost no impact on the results; compare Figure 6.13 (left) with Figure 6.13 (right). This justifies the reuse of the coarse basis throughout the simulation process to save setup time.

Thirdly, we test the weak scaling behavior; see Table 6.8. Due to the increasing number of processor cores, the mesh resolution increases and in turn also the CFL number. For 4 608 processor cores, the CFL number is still moderate with $c_{\mathrm{CFL}} = 3.6$. Based on the results in Figure 6.13, this should not impact the average iteration count substantially. All preconditioners, except for SIMPLEC, perform well. The results for 4 608 cores and SIMPLEC were not computed due to the high computational cost.

Due to the low number of iterations achieved with the monolithic preconditioner and GDSW⋆-RGDSW, this combination delivers by far the lowest solve time. Consequently, with respect to total time, it performed best.

6.4.2. Test case 2: realistic artery—transient problem. To test the preconditioners for a range of inflow velocities, we adjust the time-step size accordingly to obtain a fairly constant CFL number; see Table 6.2. In Figure 6.14 (left) we can observe the response of the preconditioning strategies to the maximum Reynolds number defined in (6.2). The PCD and monolithic preconditioner deliver robust results. If we take a closer look at the average

ETNA
Kent State University and
Johann Radon Institute (RICAM)

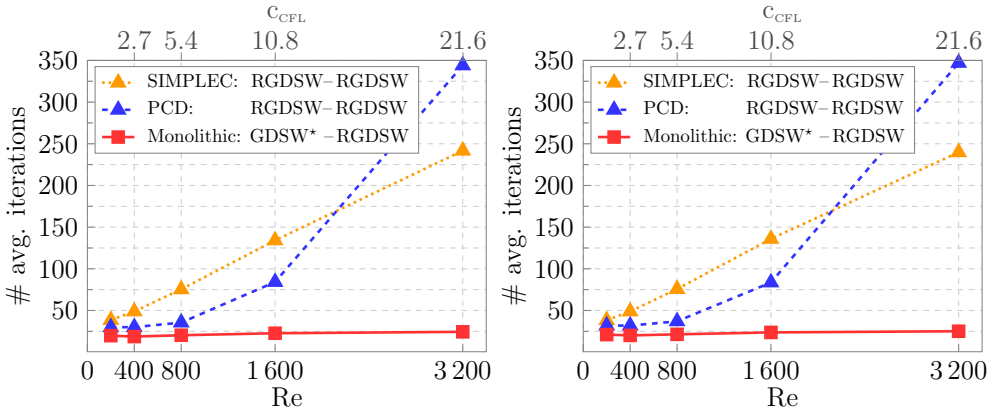

FIG. 6.13. Comparison of block and monolithic preconditioners for an increasing Reynolds and CFL number. *Use of GDSW⋆ and RGDSW coarse spaces for the velocity (first component) and pressure (second component). Transient BFS problem with P2-P1 discretization and $H/h = 9$. Increasing Reynolds number due to increasing velocity. No pressure projection is used. Exclude off-diagonal blocks in ϕ for monolithic preconditioner. Constant time-step size $\Delta t = 0.05\,\mathrm{s}$, simulation until $t = 10\,\mathrm{s}$. CFL condition increases with increasing velocity. Boundary condition strategy of PCD is (BC-3). Average GMRES iteration count per Newton step per time step. Computations of all methods with reuse of coarse basis (CB) (Left) and without reuse (Right); see Section 6.2.1. Note that the two plots look identical since the reuse of the coarse basis (CB) has a very minor effect on the iteration count in this test setting.*

TABLE 6.8
Comparison of weak scaling between block and monolithic preconditioners. *Transient BFS problem with P2-P1 discretization and $H/h = 9$. $Re = 3\,200$ with $\nu = 6.25 \times 10{-}4\,\mathrm{cm}^2/\mathrm{s}$ and inflow velocity $v_{\max} = 1\,\mathrm{cm/s}$. Average number of Newton iterations per time step required to reach convergence is 3. Due to the increasing number of subdomains, the mesh resolution increases and in turn also the CFL number c_{CFL}. No pressure projection is used. PCD boundary condition strategy is (BC-3). SIMPLEC and PCD use the RGDSW-RGDSW coarse-space combination. Exclude off-diagonal blocks in ϕ in monolithic preconditioner. Detailed results for 243 cores and varying Reynolds numbers are given in Table C.1. The results for 4 608 cores and SIMPLEC were not computed due to the high computational cost. In the case of 4 608 processor cores—to achieve the best results—8 cores are used to solve the coarse problems for PCD, 12 for the monolithic preconditioner and RGDSW-RGDSW, and 16 for the monolithic preconditioner and GDSW⋆-RGDSW.*

			Block preconditioner		Monolithic preconditioner	
# cores	c_{CFL}		SIMPLEC	PCD	RGDSW-RGDSW	GDSW⋆-RGDSW
243	1.4	# avg. iter.	52.7	31.7	35.0	21.7
		Setup time	595 s	580 s	785 s	787 s
		Solve time	2 901 s	1 761 s	2 008 s	1 268 s
		Total	3 496 s	2 341 s	2 793 s	**2 055** s
1 125	2.3	# avg. iter.	96.8	41.5	38.4	20.5
		Setup time	709 s	684 s	904 s	935 s
		Solve time	6 882 s	2 924 s	2 788 s	1 512 s
		Total	7 591 s	3 608 s	3 692 s	**2 447** s
4 608	3.6	# avg. iter.	—	43.7	43.0	19.0
		Setup time	—	745 s	994 s	1 179 s
		Solve time	—	3 567 s	3 642 s	1 664 s
		Total	—	4 312 s	4 636 s	**2 843** s

ETNA
Kent State University and
Johann Radon Institute (RICAM)

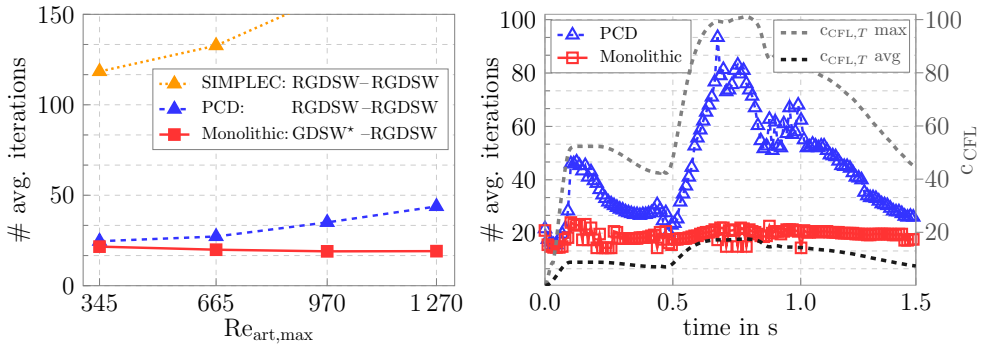

FIG. 6.14. Comparison of block and monolithic preconditioners for the transient realistic artery problem (Figure 6.3) by varying the Reynolds and CFL number. *Transient problem with P2-P1 discretization and 4.1 million degrees of freedom. Average number of Newton iterations per time step required to reach convergence is 3. Computations on 416 processor cores. No pressure projection is used. Use of GDSW⋆ and RGDSW coarse spaces for the velocity (first component) and pressure (second component). Excluding off-diagonal blocks of ϕ in monolithic preconditioner. Boundary condition strategy of PCD is (BC-3). Left: Increasing Reynolds number due to increasing initial velocity $v_{\max}$ from* 10 *to* 40 $\mathrm{cm/s}$*; see Table 6.2. Right: Configuration with* $Re_{art,\max} = 1\,270$*; see Table 6.2. Average GMRES iteration count per Newton step for each time step. Value is plotted every tenth time step. The maximum elementwise CFL number* $c_{CFL}(t)$ *is shown (light gray); its maximum over all time steps is* $c_{CFL} = 100$*. The average of elementwise CFL numbers* $c_{CFL,avg}(t)$ *is depicted as well (dark gray); its average over time is* 11*. Due to the high average GMRES iteration count, the SIMPLEC preconditioner is excluded from the figure.*

number of iterations per Newton step for the different time steps, then the monolithic approach is very robust to the velocity changes since the iteration count over time is almost constant in Figure 6.14 (right). The graph of the PCD preconditioner is similar to the graph of the CFL number; that is, it shows that its performance deteriorates with a larger CFL number.

Finally, we display strong scaling results for the artery test case in Figure 6.15; the mesh has 4.1 million degrees of freedom. Between 52 and 416 processor cores are used, of which 2 to 8 are used to solve the coarse problem; see Table 6.9. The unstructured mesh partition is constructed with METIS. We use an initial velocity of $40\,\mathrm{cm/s}$ but increase the time step with respect to Table 6.2 to $\Delta t = 0.002\,\mathrm{s}$ to reach results for 52 processor cores within a reasonable time frame. We test the monolithic preconditioner with GDSW⋆-RGDSW and the PCD block preconditioner with RGDSW-RGDSW. The results show strong scaling of both the PCD and the monolithic preconditioner. However, the monolithic preconditioner is more robust and requires significantly fewer iterations to converge, which results in a substantially lower computing time.

Based on the results in this work and the considered inflow-outflow problems, choosing the monolithic two-level overlapping Schwarz preconditioner with the GDSW⋆-RGDSW coarse-space combination provides a consistently robust performance with a low computation time. The PCD block preconditioner can achieve similarly reliable results, provided the boundary conditions of the preconditioner are chosen appropriately and the CFL number is not too high; cf., e.g., Figure 6.14.

Acknowledgments. Financial funding from the Deutsche Forschungsgemeinschaft (DFG) through the Priority Program 2311 “Robust coupling of continuum-biomechanical in silico models to establish active biological system models for later use in clinical applications—Co-design of modeling, numerics and usability”, project ID 465228106, is greatly appreciated.

ETNA
Kent State University and
Johann Radon Institute (RICAM)

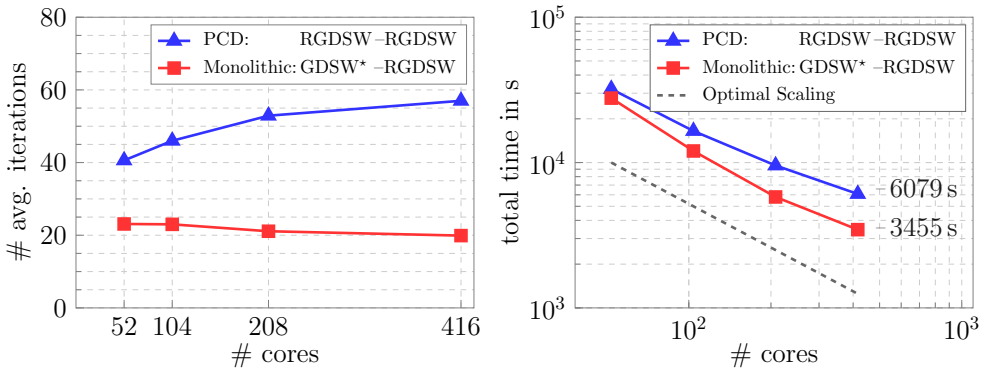

FIG. 6.15. Comparison of strong scaling of block and monolithic preconditioners for the transient realistic artery problem (Figure 6.3). *Use of GDSW⋆ and RGDSW coarse spaces for the velocity (first component) and pressure (second component). Transient problem with P2-P1 discretization and 4.1 million degrees of freedom. Average number of Newton iterations per time step required to reach convergence is 3. Computations on 52 to 416 processor cores; see Table 6.9 for the number of cores used to solve the coarse problem. Time step size* $\Delta t = 0.002\,\mathrm{s}$*, initial maximum velocity of* $40\,\mathrm{cm/s}$*, resulting in* $Re_{art,\max} = 1\,270$*. The maximum CFL number is* $c_{CFL} = 160$*. For the average elementwise CFL number, we have* $c_{CFL,avg}(t) = 28$*. The average of* $c_{CFL,avg}(t)$ *over all time steps is* 18*. No pressure projection is used. Excluding off-diagonal blocks in* ϕ*. Boundary condition strategy of PCD is (BC-3). Left: Comparison of average GMRES iteration count per Newton step per time step for different numbers of processor cores and different preconditioning techniques. Right: Total time consisting of setup and solve time.*

TABLE 6.9
Number of processor cores assigned to the coarse problem for the results in Figure 6.15. *Number of coarse functions in parentheses. Coarse spaces: The monolithic preconditioner uses GDSW⋆-RGDSW and the PCD block preconditioner RGDSW-RGDSW.*

# cores	52	104	208	416
PCD	2 (404)	3 (1 084)	4 (2 771)	8 (6 057)
Monolithic	2 (1 013)	4 (2 518)	6 (5 912)	12 (12 975)

The authors gratefully acknowledge the scientific support and HPC resources provided by the Erlangen National High Performance Computing Center (NHR@FAU) of the Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU) under the NHR project k105be. NHR funding is provided by federal and Bavarian state authorities. NHR@FAU hardware is partially funded by the German Research Foundation (DFG)-440719683.

Appendix A. The stationary Navier–Stokes problem. In Figure A.1, results for a stationary, three-dimensional lid-driven cavity problem are displayed to ensure that the results seen before in Figure 6.5 are not particularly problem specific: the results for the monolithic preconditioner, different combinations of coarse spaces, and the use of the pressure projection (4.7) qualitatively show the same behavior as before for the stationary backward-facing step problem.

The setup uses the classic formulation, where the entire top part of a unit cube is moved at constant velocity in one direction; see, for example, [40, Section 2.1]. We use a structured mesh of the unit cube $[0,1]^3$, decomposed into cubic subdomains. The subdomain resolution is $H/h = 9$, giving for a P2-P1 discretization 4.1 million degrees of freedom if $6^3 = 216$ subdomains are used and 286.7 million degrees of freedom if $25^3 = 15\,625$ subdomains are used. The kinematic viscosity is $\nu = 0.005$, giving $\mathrm{Re} = 200$. The lid includes its boundary points (classic lid-driven cavity formulation). The Dirichlet condition for the velocity at the lid is $(1,0,0)$ in each node and zero elsewhere.

ETNA
Kent State University and
Johann Radon Institute (RICAM)

It was not necessary to ensure uniqueness of the pressure by fixing a pressure node or introducing a projection into GMRES; note that prescribing a zero mean integral via a Lagrange multiplier can be computationally inefficient in a parallel setting, as the associated vector in the system matrix is dense and couples globally across all subdomains.

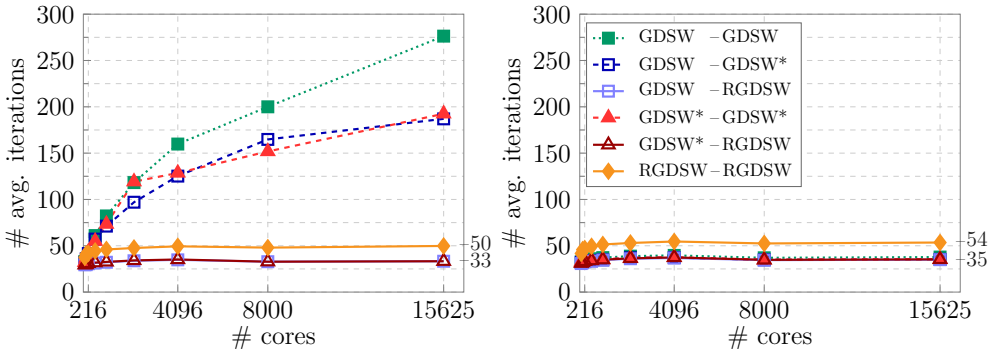

FIG. A.1. Comparison of different coarse-space combinations for the velocity (first component) and pressure (second component) for P2-P1 discretization for the monolithic preconditioner. *Average GMRES iteration count per Newton step. Weak scaling test. Stationary lid-driven cavity problem with $H/h = 9$; see Appendix A for a description. $Re = 200$. Using pressure projection in first level of monolithic preconditioner. See Figure 6.5 for corresponding BFS problem. Left: Excluding off-diagonal blocks in ϕ to build the coarse matrix. Right: Using full ϕ to build the coarse matrix.*

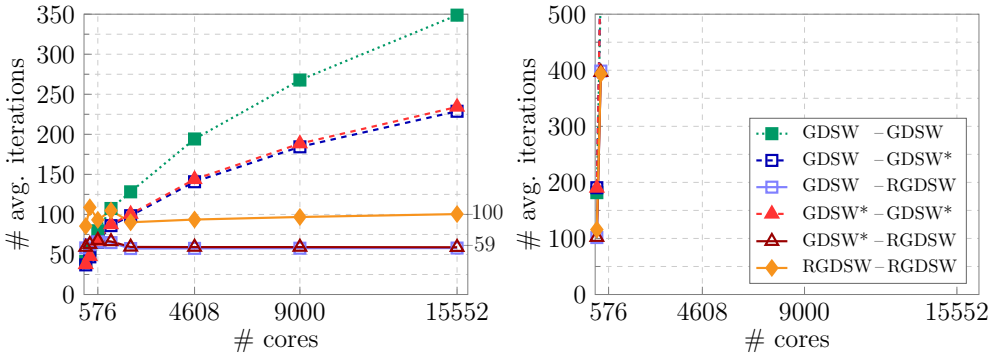

FIG. A.2. Comparison of different coarse-space combinations for the velocity (first component) and pressure (second component) for a P2-P1 discretization and the monolithic preconditioner. *Average GMRES iteration count per Newton step. Weak scaling test. Stationary BFS problem with $H/h = 9$. $Re = 200$. Omitting pressure projection; see Figure 6.5 for the corresponding results if the pressure projection is used. Left: Excluding off-diagonal blocks in ϕ to build the coarse matrix. Right: Using full ϕ to build the coarse matrix.*

ETNA
Kent State University and
Johann Radon Institute (RICAM)

TABLE A.1
Comparison of different coarse spaces for the PCD block preconditioner with different Reynolds numbers for P2-P1 discretization. *Combinations of different coarse spaces for the velocity (first component) and Schur complement components (second component). Stationary BFS problem with $H/h = 9$. (BC-2) is applied in PCD. 1 125 cores. Reynolds number is increased by decreasing the viscosity. For the definition of the Reynolds number, see (6.1). Average GMRES iteration count per Newton step and number of Newton steps in parentheses. Total time consists of setup and solve time.*

Coarse space	Velocity Pressure	GDSW GDSW	GDSW⋆ GDSW⋆	GDSW⋆ RGDSW	RGDSW RGDSW
Re = 20	# avg. iterations	76 (4)	79 (4)	76 (4)	82 (4)
	Setup time	15.0 s	13.7 s	15.0 s	12.9 s
	Solve time	35.3 s	35.1 s	33.8 s	35.2 s
	Total	50.3 s	48.8 s	48.8 s	48.1 s
Re = 200	# avg. iterations	98 (5)	100 (5)	102 (5)	103 (5)
	Setup time	15.5 s	14.2 s	13.9 s	12.9 s
	Solve time	57.7 s	56.7 s	57.2 s	56.9 s
	Total	73.2 s	70.9 s	70.1 s	69.8 s
Re = 400	# avg. iterations	163 (6)	167 (6)	174 (6)	173 (6)
	Setup time	15.9 s	13.8 s	13.7 s	13.0 s
	Solve time	122.3 s	120.2 s	125.0 s	119.8 s
	Total	138.2 s	134.0 s	138.7 s	132.8 s

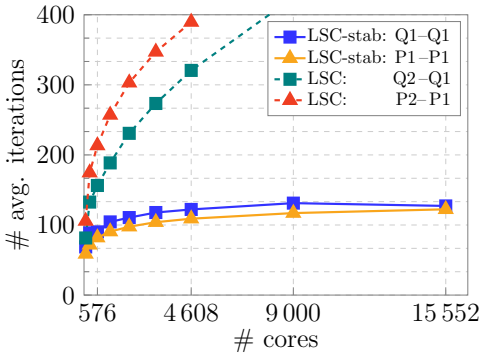

FIG. A.3. Comparison of different finite element discretizations for the LSC block-triangular preconditioner. *Use of RGDSW-RGDSW coarse space for the velocity (first component) and for Schur complement components (second component). Average GMRES iteration count per Newton step. Total number of Newton iterations required to reach convergence is 5. Weak scaling test. Stationary BFS problem with configurations according to Table 6.1. Re = 200. Use of Bochev–Dohrmann stabilization for P1-P1 and Q1-Q1 elements. No use of pressure projection. LSC block preconditioner; see Section 3.2.*

ETNA
Kent State University and
Johann Radon Institute (RICAM)

Appendix B. The transient Navier–Stokes problem.

TABLE B.1
Comparison of different finite element discretizations for the monolithic preconditioner for an increasing Reynolds number. *Average GMRES iteration count per Newton step per over all time steps. Transient BFS problem with configurations according to Table 6.1. Time step size is constant, $\Delta t = 0.05\,\mathrm{s}$, simulation until $t = 10\,\mathrm{s}$, which amounts to 200 time steps in total. Use of Bochev–Dohrmann stabilization for P1-P1 and Q1-Q1 elements. 243 processor cores. Combinations of different coarse spaces for the velocity (first component) and pressure (second component). No use of pressure projection. Exclude off-diagonal blocks in ϕ. Results corresponding to Figure 6.8.*

Discretization			P1-P1	P2-P1	Q1-Q1	Q2-Q1	Q2-P1-disc.
Coarse space		Velocity Pressure	RGDSW RGDSW	GDSW* RGDSW	RGDSW RGDSW	GDSW* RGDSW	GDSW* GDSW*
Re	ν	$v_{\max}$	# avg. iter. per Newton step over all time steps				
200	0.01	1	13.7	19.8	12.8	21.8	31.7
400	0.005	1	13.8	18.7	14.3	19.8	35.3
800	0.0025	1	14.9	19.5	14.9	20.7	37.3
1 600	0.00125	1	14.9	20.8	14.7	22.6	39.1
3 200	0.000625	1	15.5	21.7	13.4	20.3	40.7
200	0.01	1	13.7	19.8	12.8	21.8	31.7
400	0.01	2	12.8	18.7	12.3	20.0	30.7
800	0.01	4	13.1	20.3	13.6	22.0	31.7
1 600	0.01	8	14.5	22.7	14.5	23.6	33.6
3 200	0.01	16	24.9	24.4	24.3	27.5	39.9

TABLE B.2
Comparison of different boundary condition strategies for the PCD block preconditioner and a transient simulation. *Boundary conditions in PCD are defined according to Section 3.1.2. Transient BFS problem with P2-P1 discretization. Average iteration count per Newton step over all time steps, number of Newton steps in parentheses. GDSW-GDSW coarse space for the velocity (first component) and pressure (second component). Number of processor cores is 243. Time step $\Delta t = 0.02\,\mathrm{s}$, simulation until $t = 1.0\,\mathrm{s}$, which amounts to 50 time steps. (*) Results corresponding to Figure 6.10 (left).*

Boundary condition strategy	(BC-1)	(BC-2)	(BC-3) (*)
# avg. iterations	33.5 (3.1)	33.4 (3.1)	33.4 (3.1)

ETNA
Kent State University and
Johann Radon Institute (RICAM)

Appendix C. Comparison of block and monolithic preconditioners for varying Reynolds numbers.

TABLE C.1
Comparison of block and monolithic preconditioners for an increasing Reynolds number. *Average GMRES iteration count per Newton step per time step. Total time in seconds consisting of setup and solve time. Transient BFS problem with P2-P1 discretization and $H/h = 9$. Increasing Reynolds number due to decreasing viscosity. Computation of 200 time steps with time-step size $\Delta t = 0.05$ s, simulation until $t = 10$ s. CFL number is constant and equal to 1.35. Computations on 243 cores. Boundary condition strategy in PCD is (BC-3). Use of GDSW* and RGDSW coarse spaces. Results corresponding to Figure 6.12.*

	Re	200	400	800	1 600	3 200
	ν	$1.0{\cdot}10^{-2}$	$5.0{\cdot}10^{-3}$	$2.5{\cdot}10^{-3}$	$1.3{\cdot}10^{-3}$	$6.25{\cdot}10^{-4}$
SIMPLEC RGDSW-RGDSW	# avg. iter.	38.6 (3.0)	39.0 (3.0)	41.1 (3.0)	46.7 (3.0)	52.7 (3.0)
	Setup time	596 s	594 s	598 s	599 s	595 s
	Solve time	2 089 s	2 116 s	2 298 s	2 570 s	2 901 s
	Total	2 685 s	2 710 s	2 896 s	3 169 s	3 496 s
PCD RGDSW-RGDSW	# avg. iter.	29.9 (2.8)	30.7 (3.0)	28.9 (3.0)	28.8 (3.0)	31.7 (3.0)
	Setup time	584 s	578 s	577 s	581 s	580 s
	Solve time	1 592 s	1 700 s	1 629 s	1 600 s	1 761 s
	Total	2 176 s	2 278 s	2 206 s	2 181 s	2 341 s
Monolithic RGDSW-RGDSW	# avg. iter.	26.9 (2.8)	29.2 (3.0)	30.5 (3.0)	32.7 (3.0)	35.0 (3.0)
	Setup time	795 s	789 s	789 s	785 s	785 s
	Solve time	1 458 s	1 671 s	1 747 s	1 873 s	2 008 s
	Total	2 253 s	2 460 s	2 536 s	2 658 s	2 793 s
Monolithic GDSW*-RGDSW	# avg. iter.	19.8 (2.8)	18.7 (2.8)	19.5 (2.9)	20.8 (3.0)	21.7 (3.1)
	Setup time	790 s	783 s	786 s	782 s	787 s
	Solve time	1 074 s	1 035 s	1 122 s	1 203 s	1 268 s
	Total	**1 864** s	**1 818** s	**1 908** s	**1 984** s	**2 055** s

REFERENCES

[1] D. BALZANI, D. BÖSE, D. BRANDS, R. ERBEL, A. KLAWONN, O. RHEINBACH, AND J. SCHRÖDER, *Parallel simulation of patient-specific atherosclerotic arteries for the enhancement of intravascular ultrasound diagnostics*, Eng. Comput., 29 (2012), pp. 888–906.

[2] D. BALZANI, S. DEPARIS, S. FAUSTEN, D. FORTI, A. HEINLEIN, A. KLAWONN, A. QUARTERONI, O. RHEINBACH, AND J. SCHRÖDER, *Numerical modeling of fluid-structure interaction in arteries with anisotropic polyconvex hyperelastic and anisotropic viscoelastic material models at finite strains*, Int. J. Numer. Methods Biomed. Eng., 32 (2016), Paper No. e02756, 41 pages.

[3] D. BALZANI, A. HEINLEIN, A. KLAWONN, J. KNEPPER, S. N. RAMESH, O. RHEINBACH, L. SASSMANNSHAUSEN, AND K. UHLMANN, *A computational framework for pharmaco-mechanical interactions in arterial walls using parallel monolithic domain decomposition methods*, GAMM-Mitt., 47 (2024), Paper No. e202370002, 26 pages.

[4] E. BAVIER, M. HOEMMEN, S. RAJAMANICKAM, AND H. THORNQUIST, *Amesos2 and Belos: direct and iterative solvers for large sparse linear systems*, Sci. Program., 20 (2012), pp. 241–255.

[5] M. BENZI, M. NG, Q. NIU, AND Z. WANG, *A relaxed dimensional factorization preconditioner for the incompressible Navier-Stokes equations*, J. Comput. Phys., 230 (2011), pp. 6185–6202.

ETNA
Kent State University and
Johann Radon Institute (RICAM)

[6] M. BENZI AND Z. WANG, *A parallel implementation of the modified augmented Lagrangian preconditioner for the incompressible Navier-Stokes equations*, Numer. Algorithms, 64 (2013), pp. 73–84.
[7] T. BEVILACQUA, F. DASSI, S. ZAMPINI, AND S. SCACCHI, *BDDC preconditioners for virtual element approximations of the three-dimensional Stokes equations*, SIAM J. Sci. Comput., 46 (2024), pp. A156–A178.
[8] D. BOFFI, F. BREZZI, AND M. FORTIN, *Mixed Finite Element Methods and Applications*, Springer, Berlin, 2013.
[9] N. BOOTLAND, A. BENTLEY, C. KEES, AND A. WATHEN, *Preconditioners for two-phase incompressible Navier-Stokes flow*, SIAM J. Sci. Comput., 41 (2019), pp. B843–B869.
[10] X.-C. CAI AND M. SARKIS, *A restricted additive Schwarz preconditioner for general sparse linear systems*, SIAM J. Sci. Comput., 21 (1999), pp. 792–797.
[11] J.-M. CROS, *A preconditioner for the Schur complement domain decomposition method*, in Domain Decomposition Methods in Science and Engineering, I. Herrera, D. Keyes, O. Widlund, and R. Yates, eds., National Autonomous University of Mexico, Mexico City, 2003, pp. 373–380.
[12] E. C. CYR, J. N. SHADID, AND R. S. TUMINARO, *Stabilization and scalable block preconditioning for the Navier-Stokes equations*, J. Comput. Phys., 231 (2012), pp. 345–363.
[13] ———, *Teko: a block preconditioning capability with concrete example applications in Navier-Stokes and MHD*, SIAM J. Sci. Comput., 38 (2016), pp. S307–S331.
[14] S. DEPARIS, G. GRANDPERRIN, AND A. QUARTERONI, *Parallel preconditioners for the unsteady Navier-Stokes equations and applications to hemodynamics simulations*, Comput. & Fluids, 92 (2014), pp. 253–273.
[15] C. R. DOHRMANN, *A preconditioner for substructuring based on constrained energy minimization*, SIAM J. Sci. Comput., 25 (2003), pp. 246–258.
[16] C. R. DOHRMANN AND P. B. BOCHEV, *A stabilized finite element method for the Stokes problem based on polynomial pressure projections*, Internat. J. Numer. Methods Fluids, 46 (2004), pp. 183–201.
[17] C. R. DOHRMANN, A. KLAWONN, AND O. B. WIDLUND, *Domain decomposition for less regular subdomains: overlapping Schwarz in two dimensions*, SIAM J. Numer. Anal., 46 (2008), pp. 2153–2168.
[18] ———, *A family of energy minimizing coarse spaces for overlapping Schwarz preconditioners*, in Domain Decomposition Methods in Science and Engineering XVII, vol. 60 of Lect. Notes Comput. Sci. Eng., Springer, Berlin, 2008, pp. 247–254.
[19] C. R. DOHRMANN AND O. B. WIDLUND, *An overlapping Schwarz algorithm for almost incompressible elasticity*, SIAM J. Numer. Anal., 47 (2009), pp. 2897–2923.
[20] ———, *On the design of small coarse spaces for domain decomposition algorithms*, SIAM J. Sci. Comput., 39 (2017), pp. A1466–A1488.
[21] J. P. VAN DOORMAAL AND G. D. RAITHBY, *Enhancements of the simple method for predicting incompressible fluid flows*, Numer. Heat Transf., 7 (1984), pp. 147–163.
[22] H. C. EDWARDS, C. R. TROTT, AND D. SUNDERLAND, *Kokkos: Enabling manycore performance portability through polymorphic memory access patterns*, J. Parallel Distrib. Comput., 74 (2014), pp. 3202–3216.
[23] S. C. EISENSTAT AND H. F. WALKER, *Globally convergent inexact Newton methods*, SIAM J. Optim., 4 (1994), pp. 393–422.
[24] S. C. EISENSTAT AND H. F. WALKER, *Choosing the forcing terms in an inexact Newton method*, SIAM J. Sci. Comput., 17 (1996), pp. 16–32.
[25] H. ELMAN, V. E. HOWLE, J. SHADID, R. SHUTTLEWORTH, AND R. TUMINARO, *Block preconditioners based on approximate commutators*, SIAM J. Sci. Comput., 27 (2006), pp. 1651–1668.
[26] H. ELMAN, V. E. HOWLE, J. SHADID, R. SHUTTLEWORTH, AND R. TUMINARO, *A taxonomy and comparison of parallel block multi-level preconditioners for the incompressible Navier-Stokes equations*, J. Comput. Phys., 227 (2008), pp. 1790–1808.
[27] H. ELMAN, V. E. HOWLE, J. SHADID, D. SILVESTER, AND R. TUMINARO, *Least squares preconditioners for stabilized discretizations of the Navier-Stokes equations*, SIAM J. Sci. Comput., 30 (2007/08), pp. 290–311.
[28] H. ELMAN AND D. SILVESTER, *Fast nonsymmetric iterations and preconditioning for Navier-Stokes equations*, SIAM J. Sci. Comput., 17 (1996), pp. 33–46.
[29] H. ELMAN, D. SILVESTER, AND A. WATHEN, *Finite Elements and Fast Iterative Solvers: With Applications in Incompressible Fluid Dynamics*, 2nd ed., Oxford University Press, Oxford, 2014.
[30] H. C. ELMAN, *Preconditioning for the steady-state Navier-Stokes equations with low viscosity*, SIAM J. Sci. Comput., 20 (1999), pp. 1299–1316.
[31] H. C. ELMAN AND R. S. TUMINARO, *Boundary conditions in approximate commutator preconditioners for the Navier-Stokes equations*, Electron. Trans. Numer. Anal., 35 (2009), pp. 257–280. https://etna.ricam.oeaw.ac.at/vol.35.2009/pp257-280.dir/pp257-280.pdf
[32] C. FARHAT, M. LESOINNE, P. LETALLEC, K. PIERSON, AND D. RIXEN, *FETI-DP: a dual–primal unified FETI method—part I: a faster alternative to the two-level FETI method*, Internat. J. Numer. Methods Engrg., 50 (2001), pp. 1523–1544.

ETNA
Kent State University and
Johann Radon Institute (RICAM)

[33] C. FARHAT, M. LESOINNE, AND K. PIERSON, *A scalable dual-primal domain decomposition method*, Numer. Linear Algebra Appl., 7 (2000), pp. 687–714.
[34] FEDDLIB TEAM, *FEDDLib (Finite Element and Domain Decomposition Library)*, GitHub repository, 2025. https://github.com/FEDDLib/FEDDLib
[35] A. GAUTHIER, F. SALERI, AND A. VENEZIANI, *A fast preconditioner for the incompressible Navier-Stokes equations*, Comput. Vis. Sci., 6 (2004), pp. 105–112.
[36] B. E. GRIFFITH, *An accurate and efficient method for the incompressible Navier-Stokes equations using the projection method as a preconditioner*, J. Comput. Phys., 228 (2009), pp. 7565–7595.
[37] E. HAIRER AND G. WANNER, *Solving Ordinary Differential Equations. II.*, Springer, Berlin, 1996.
[38] M. HANEK, J. ŠÍSTEK, AND P. BURDA, *Multilevel BDDC for incompressible Navier-Stokes equations*, SIAM J. Sci. Comput., 42 (2020), pp. C359–C383.
[39] X. HE, C. VUIK, AND C. KLAIJ, *Block-preconditioners for the incompressible Navier-Stokes equations discretized by a finite volume method*, J. Numer. Math., 25 (2017), pp. 89–105.
[40] A. HEINLEIN, C. HOCHMUTH, AND A. KLAWONN, *Monolithic overlapping Schwarz domain decomposition methods with GDSW coarse spaces for incompressible fluid flow problems*, SIAM J. Sci. Comput., 41 (2019), pp. C291–C316.
[41] ———, *Reduced dimension GDSW coarse spaces for monolithic Schwarz domain decomposition methods for incompressible fluid flow problems*, Internat. J. Numer. Methods Engrg., 121 (2020), pp. 1101–1119.
[42] ———, *Fully algebraic two-level overlapping Schwarz preconditioners for elasticity problems*, in Numerical Mathematics and Advanced Applications—ENUMATH 2019, vol. 139 of Lect. Notes Comput. Sci. Eng., Springer, Cham, 2021, pp. 531–539.
[43] A. HEINLEIN, A. KLAWONN, J. KNEPPER, O. RHEINBACH, AND O. B. WIDLUND, *Adaptive GDSW coarse spaces of reduced dimension for overlapping Schwarz methods*, SIAM J. Sci. Comput., 44 (2022), pp. A1176–A1204.
[44] A. HEINLEIN, A. KLAWONN, S. RAJAMANICKAM, AND O. RHEINBACH, *FROSch: a fast and robust overlapping Schwarz domain decomposition preconditioner based on Xpetra in Trilinos*, in Domain Decomposition Methods in Science and Engineering XXV, vol. 138 of Lect. Notes Comput. Sci. Eng., Springer, Cham, 2020, pp. 176–184.
[45] A. HEINLEIN, A. KLAWONN, AND O. RHEINBACH, *A parallel implementation of a two-level overlapping Schwarz method with energy-minimizing coarse space based on Trilinos*, SIAM J. Sci. Comput., 38 (2016), pp. C713–C747.
[46] A. HEINLEIN, A. KLAWONN, O. RHEINBACH, AND O. B. WIDLUND, *Improving the parallel performance of overlapping Schwarz methods by using a smaller energy minimizing coarse space*, in Domain Decomposition Methods in Science and Engineering XXIV, vol. 125 of Lect. Notes Comput. Sci. Eng., Springer, Cham, 2018, pp. 383–392.
[47] A. HEINLEIN, O. RHEINBACH, AND F. RÖVER, *A multilevel extension of the GDSW overlapping Schwarz preconditioner in two dimensions*, Comput. Methods Appl. Math., 23 (2023), pp. 953–968.
[48] ———, *Parallel scalability of three-level FROSch preconditioners to 220000 cores using the theta supercomputer*, SIAM J. Sci. Comput., 45 (2023), pp. S173–S198.
[49] M. A. HEROUX, R. A. BARTLETT, V. E. HOWLE, R. J. HOEKSTRA, J. J. HU, T. G. KOLDA, R. B. LEHOUCQ, K. R. LONG, R. P. PAWLOWSKI, E. T. PHIPPS, A. G. SALINGER, H. K. THORNQUIST, R. S. TUMINARO, J. M. WILLENBRING, A. WILLIAMS, AND K. S. STANLEY, *An overview of the Trilinos Project*, ACM Trans. Math. Software, 31 (2005), pp. 397–423.
[50] C. HOCHMUTH, *Parallel Overlapping Schwarz Preconditioners for Incompressible Fluid Flow and Fluid-Structure Interaction Problems*, PhD. Thesis, University of Cologne, Cologne, 2020.
[51] V. E. HOWLE, J. SCHRODER, AND R. TUMINARO, *The effect of boundary conditions within Pressure Convection–Diffusion preconditioners*, Tech. Rep. SAND2006-4466, Sandia National Laboratories, Albuquerque and Livermore, 2006.
[52] G. KARYPIS AND V. KUMAR, *A fast and high quality multilevel scheme for partitioning irregular graphs*, SIAM J. Sci. Comput., 20 (1998), pp. 359–392.
[53] D. KAY, D. LOGHIN, AND A. WATHEN, *A preconditioner for the steady-state Navier-Stokes equations*, SIAM J. Sci. Comput., 24 (2002), pp. 237–256.
[54] A. KLAWONN, *Block-triangular preconditioners for saddle point problems with a penalty term*, SIAM J. Sci. Comput., 19 (1998), pp. 172–184.
[55] ———, *An optimal preconditioner for a class of saddle point problems with a penalty term*, SIAM J. Sci. Comput., 19 (1998), pp. 540–552.
[56] A. KLAWONN AND M. LANSER, *Nonlinear monolithic two-level Schwarz methods for the Navier-Stokes equations*, Preprint on arXiv, 2024. https://arxiv.org/abs/2409.03041
[57] A. KLAWONN AND L. F. PAVARINO, *Overlapping Schwarz methods for mixed linear elasticity and Stokes problems*, Comput. Methods Appl. Mech. Engrg., 165 (1998), pp. 233–245.
[58] ———, *A comparison of overlapping Schwarz methods and block preconditioners for saddle point problems*, Numer. Linear Algebra Appl., 7 (2000), pp. 1–25.

ETNA
Kent State University and
Johann Radon Institute (RICAM)

[59] A. KLAWONN AND G. STARKE, *Block triangular preconditioners for nonsymmetric saddle point problems: field-of-values analysis*, Numer. Math., 81 (1999), pp. 577–594.
[60] J. KNEPPER, *Adaptive Coarse Spaces for the Overlapping Schwarz Method and Multiscale Elliptic Problems*, PhD. Thesis, University of Cologne, Cologne, 2022.
[61] J. KORELC AND P. WRIGGERS, *Automation of Finite Element Methods*, Springer, Cham, 2016.
[62] L. LAUDER, S. EWEN, A. R. TZAFRIRI, E. R. EDELMAN, T. F. LÜSCHER, P. J. BLANKESTIJN, O. DÖRR, M. P. SCHLAICH, F. SHARIF, M. VOSKUIL, T. ZELLER, C. UKENA, B. SCHELLER, M. BÖHM, AND F. MAHFOUD, *Renal artery anatomy assessed by quantitative analysis of selective renal angiography in 1,000 patients with hypertension*, EuroIntervention, 14 (2018), pp. 121–128.
[63] J. LI AND O. WIDLUND, *BDDC algorithms for incompressible Stokes equations*, SIAM J. Numer. Anal., 44 (2006), pp. 2432–2455.
[64] Z.-Z. LIANG AND G.-F. ZHANG, *SIMPLE-like preconditioners for saddle point problems from the steady Navier-Stokes equations*, J. Comput. Appl. Math., 302 (2016), pp. 211–223.
[65] M. MAYR, A. HEINLEIN, C. GLUSA, S. RAJAMANICKAM, M. ARNST, R. BARTLETT, L. BERGER-VERGIAT, E. BOMAN, K. DEVINE, G. HARPER, M. HEROUX, M. HOEMMEN, J. HU, B. KELLEY, K. KIM, D. P. KOURI, P. KUBERRY, K. LIEGEOIS, C. C. OBER, R. PAWLOWSKI, C. PEARSON, M. PEREGO, E. PHIPPS, D. RIDZAL, N. V. ROBERTS, C. SIEFERT, H. THORNQUIST, R. TOMASETTI, C. R. TROTT, R. S. TUMINARO, J. M. WILLENBRING, M. M. WOLF, AND I. YAMAZAKI, *Trilinos: Enabling scientific computing across diverse hardware architectures at scale*, Preprint on arXiv, 2025. https://arxiv.org/abs/2503.08126
[66] NATIONAL INSTITUTE OF SCIENCE AND TECHNOLOGY IN MEDICINE ASSISTED BY SCIENTIFIC COMPUTING (BRAZIL), *HeMoLab*, 2014. Website Accessed November 2014. http://hemolab.lncc.br
[67] S. PATANKAR AND D. SPALDING, *A calculation procedure for heat, mass and momentum transfer in three-dimensional parabolic flows*, Int. J. Heat Mass Transf., 15 (1972), pp. 1787–1806.
[68] Y. PEI, Q. YE, J. XIAO, H. SHEN, X. LONG, AND X. PENG, *Relationship between blood flow velocity and blood suppression inversion time for optimal visualization of renal artery in spatial labeling with multiple inversion pulses: in vitro study*, Iran J. Radiol., 15 (2018), Paper No. e65502, 7 pages.
[69] M. PERNICE AND M. D. TOCCI, *A multigrid-preconditioned Newton-Krylov method for the incompressible Navier-Stokes equations*, SIAM J. Sci. Comput., 23 (2001), pp. 398–418.
[70] M. PERNICE AND H. F. WALKER, *NITSOL: a Newton iterative solver for nonlinear systems*, SIAM J. Sci. Comput., 19 (1998), pp. 302–318.
[71] A. QUARTERONI, F. SALERI, AND A. VENEZIANI, *Factorization methods for the numerical approximation of Navier-Stokes equations*, Comput. Methods Appl. Mech. Engrg., 188 (2000), pp. 505–526.
[72] S. RAJAMANICKAM, S. ACER, L. BERGER-VERGIAT, V. DANG, N. ELLINGWOOD, E. HARVEY, B. KELLEY, C. R. TROTT, J. WILKE, AND I. YAMAZAKI, *Kokkos Kernels: performance portable sparse/dense linear algebra and graph kernels*, Preprint on arXiv, 2021. https://arxiv.org/abs/2103.11991
[73] Y. SAAD AND M. H. SCHULTZ, *GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 856–869.
[74] A. SEGAL, M. UR REHMAN, AND C. VUIK, *Preconditioners for incompressible Navier-Stokes solvers*, Numer. Math. Theory Methods Appl., 3 (2010), pp. 245–275.
[75] D. SILVESTER, H. ELMAN, D. KAY, AND A. WATHEN, *Efficient preconditioning of the linearized Navier-Stokes equations for incompressible flow*, J. Comput. Appl. Math., 128 (2001), pp. 261–279.
[76] B. F. SMITH, P. E. BJØRSTAD, AND W. D. GROPP, *Domain Decomposition*, Cambridge University Press, Cambridge, 1996.
[77] A. TOSELLI AND O. B. WIDLUND, *Domain Decomposition Methods—Algorithms and Theory*, Springer, Berlin, 2005.
[78] TRILINOS PROJECT TEAM, *Trilinos*, GitHub repository, 2025. https://github.com/trilinos/Trilinos
[79] ———, *Trilinos Project Website*, 2025. https://trilinos.github.io
[80] C. TROTT, L. BERGER-VERGIAT, D. POLIAKOFF, S. RAJAMANICKAM, D. LEBRUN-GRANDIE, J. MADSEN, N. AL AWAR, M. GLIGORIC, G. SHIPMAN, AND G. WOMELDORFF, *The Kokkos EcoSystem: Comprehensive performance portability for high performance computing*, Comput. Sci. & Eng., 23 (2021), pp. 10–18.
[81] T. YAMAMOTO, Y. OGASAWARA, A. KIMURA, H. TANAKA, O. HIRAMATSU, K. TSUJIOKA, M. J. LEVER, K. H. PARKER, C. J. JONES, C. G. CARO, AND F. KAJIYA, *Blood velocity profiles in the human renal artery by Doppler ultrasound and their relationship to atherosclerosis*, Arterioscler. Thromb. Vasc. Biol., 16 (1996), pp. 172–177.
[82] I. YAMAZAKI, A. HEINLEIN, AND S. RAJAMANICKAM, *An experimental study of two-level Schwarz domain-decomposition preconditioners on GPUs*, in 2023 IEEE International Parallel and Distributed Processing Symposium (IPDPS), IEEE Conference Proceedings, Los Alamitos, 2023, pp. 680–689.