SIAM J. SCI. COMPUT.
Vol. 41, No. 4, pp. C291–C316

MONOLITHIC OVERLAPPING SCHWARZ DOMAIN DECOMPOSITION METHODS WITH GDSW COARSE SPACES FOR INCOMPRESSIBLE FLUID FLOW PROBLEMS*

ALEXANDER HEINLEIN†‡, CHRISTIAN HOCHMUTH†, AND AXEL KLAWONN†‡

Abstract. Monolithic overlapping Schwarz preconditioners for saddle point problems of Stokes and Navier–Stokes type are presented. In order to obtain numerically scalable algorithms, coarse spaces obtained from the generalized Dryja–Smith–Widlund (GDSW) approach are used. Numerical results of our parallel implementation are presented for various incompressible fluid flow problems. In particular, cases are considered where the problem cannot or should not be reduced using local static condensation, e.g., Stokes or Navier–Stokes problems with continuous pressure spaces. In the new monolithic preconditioners, the local overlapping problems and the coarse problem are saddle point problems with the same structure as the original problem. Our parallel implementation of these preconditioners is based on the fast and robust overlapping Schwarz (FROSch) library, which is part of the Trilinos package ShyLU. The implementation is essentially algebraic in the sense that, for the class of problems presented here, the preconditioners can be constructed from the fully assembled stiffness matrix and information about the block structure of the problem. Further information about the geometry or the null space of the underlying problem can improve the performance compared to the default settings. Parallel scalability results for several thousand cores for Stokes and Navier–Stokes model problems are reported. Each of the local problems is solved using a direct solver in serial mode, whereas the coarse problem is solved using a direct solver in serial or message passing interface (MPI)-parallel mode or using an MPI-parallel iterative Krylov solver.

Key words. saddle point problems, Stokes, Navier–Stokes, GDSW, monolithic overlapping Schwarz, algebraic preconditioner, parallel computing, domain decomposition

AMS subject classifications. 65F10, 65M55, 65Y05, 65F08

DOI. 10.1137/18M1184047

1. Introduction.

Saddle point problems arise in many physical applications, e.g., in the simulation of fluid flow. We consider finite element discretizations of the underlying partial differential equations which can result in large and ill-conditioned linear systems. In addition to the required inf-sup conditions or a suitable stabilization for finite element discretizations of saddle point problems, special care has to be taken when constructing preconditioners for the discrete problem. In particular, the block structure and the coupling blocks have to be handled appropriately to guarantee fast convergence of iterative methods.

In this paper, we propose a monolithic two-level overlapping Schwarz preconditioner with discrete harmonic coarse spaces. Specifically, we extend generalized Dryja–Smith–Widlund (GDSW) coarse spaces, which were originally introduced for

*Submitted to the journal's Software and High-Performance Computing section April 30, 2018; accepted for publication (in revised form) April 4, 2019; published electronically July 2, 2019.
https://doi.org/10.1137/18M1184047

Funding: The authors gratefully acknowledge financial support by the German Science Foundation (DFG), project KL2094/3-1 and the computing time granted by the Center for Computational Sciences and Simulation (CCSS) of the University of Duisburg-Essen and provided on the supercomputer magnitUDE (DFG grants INST 20876/209-1 FUGG, INST 20876/243-1 FUGG) at the Zentrum für Informations- und Mediendienste (ZIM).

†Department of Mathematics and Computer Science, University of Cologne, Weyertal 86-90, 50931 Köln, Germany (alexander.heinlein@uni-koeln.de, c.hochmuth@uni-koeln.de, axel.klawonn@uni-koeln.de, url: http://www.numerik.uni-koeln.de).

‡Center for Data and Simulation Science, University of Cologne, 50931 Köln, Germany (http://www.cds.uni-koeln.de).

certain elliptic problems in [19, 20], to general saddle point problems. For elliptic problems, the GDSW coarse basis functions are energy-minimal extensions representing the null space of the elliptic operator. In the previous works [21, 22] by Dohrmann and Widlund on GDSW preconditioners for saddle point problems, only discontinuous pressure spaces were considered. Consequently, the saddle point problems could be reduced to elliptic problems by static condensation of the pressure. In contrast, our method is inspired by the monolithic Schwarz preconditioners with Lagrangian coarse basis functions introduced by Klawonn and Pavarino in [50, 51], which operate on the full saddle point problem. The coarse basis functions of our new monolithic GDSW preconditioner are discrete saddle point harmonic extensions of interface functions. One significant advantage of our new method is that it can be applied to arbitrary geometries and domain decompositions, whereas the use of Lagrangian coarse basis functions requires a coarse triangulation. This limits the use of Schwarz methods with Lagrangian coarse spaces for arbitrary geometries. Preliminary results without algorithmic and implementation details for a related monolithic GDSW preconditioner for continuous pressure spaces, based on ideas of Klawonn and Pavarino in [50, 51], were presented by Clark Dohrmann at the Workshop on Adaptive Finite Elements and Domain Decomposition Methods held in 2010 in Milan, Italy; cf. [18].

In [22, 24, 25], different approaches for reducing the dimension of GDSW coarse spaces have been introduced, and some of them have been proven to significantly improve the parallel scalability compared to standard GDSW coarse spaces; cf. [42]. Alternatively, the parallel scalability could be improved by solving the coarse level inexactly using another GDSW preconditioner resulting in a three-level GDSW method; cf. [41]. For highly heterogeneous multiscale problems, GDSW coarse spaces were enhanced using local generalized eigenvalue problems in [36, 35], resulting in a method that is robust with respect to the contrast of the coefficients.

As described in [38], in a parallel implementation, GDSW coarse spaces can be constructed in an algebraic fashion directly from the fully assembled matrix. The parallel implementation of our new Schwarz preconditioners with GDSW coarse spaces for saddle point problems is based on the FROSch (fast and robust overlapping Schwarz) software [37] in the package ShyLU of the Trilinos library [43]. FROSch is a framework for Schwarz preconditioners in Trilinos, and one of its main contributions is a parallel and algebraic implementation of the GDSW preconditioner for elliptic problems; see also [33, 40, 39, 38, 42]. Therefore, the parallel implementation described here is also algebraic, i.e., the preconditioner can be easily built requiring only few input parameters, and will be described in more detail in section 7. The functionality of our new implementation has already been added to FROSch and is therefore available as open source as part of Trilinos.

In our approach, we solve saddle point problems of the form (1) using the generalized minimal residual method (GMRES). We consider the Stokes and Navier–Stokes equations in two and three dimensions. An extensive overview of numerical methods for saddle point problems is given in [6]. Older approaches for the iterative solution of those saddle point problems are, e.g., the exact and inexact Uzawa algorithm [1, 3, 11, 29, 57, 71], where velocity and pressure are decoupled and solved in a segregated approach. Other physically based approaches are the semi-implicit method for pressure linked equations (SIMPLE) and its generalizations, i.e., SIMPLEC and SIMPLER; cf. [27, 53, 54]. Block preconditioners for GMRES, MINRES, and the conjugate residual method are presented in [28, 58, 62, 64, 68, 48, 51, 49, 52]. Further block preconditioners are the pressure convection-diffusion (PCD) preconditioner [63, 47, 31], the least-squares commutator (LSC) preconditioner [26], Yosida's

method [55, 56], the relaxed dimensional factorization (RDF) preconditioner [8], and the dimensional splitting (DS) preconditioner [7, 17]. Domain decomposition methods for Stokes and Navier–Stokes problems were studied in [10, 71]. Schwarz preconditioners for saddle point problems have already been used for the approximation of the inverse matrices of blocks in [15, 2, 16, 38] and as monolithic preconditioners in [65, 14, 4, 5, 70]. Alternative solvers for saddle point problems are, e.g., multigrid methods; cf. [67, 66, 69, 9, 12, 45, 32]. Let us note that there are several other publications on iterative solvers for saddle point problems.

The paper is arranged as follows. In section 2, we present the model problems. In section 3, we describe the mixed finite elements used in our discretization. Then, we recapitulate the definition of two-level Schwarz preconditioners for elliptic problems in section 4 and explain energy-minimizing GDSW coarse spaces. In section 5, we describe monolithic two-level Schwarz preconditioners for saddle point problems. First, we introduce a monolithic Schwarz preconditioner with Lagrangian coarse space and the new monolithic GDSW preconditioner. Furthermore, we discuss how we account for a nonunique pressure in the preconditioner in section 6. Details on the parallel and algebraic implementation of the monolithic preconditioner based on the FROSch framework are given in section 7. In section 8, we present numerical results for the model problems.

2. Saddle point problems. We construct our monolithic preconditioner for the two model problems presented in this section. In the following, we assume that $\Omega \subset \mathbb{R}^d$, $d = 2, 3$, is a polygonal or polyhedral open set.

2.1. Stokes equations. Our first model problem is given by the Stokes equations in two and three dimensions. We seek to determine the velocity $u \in V_g$ with $V_g = \{v \in (H^1(\Omega))^d : v_{|\partial\Omega_D} = g\}$ and the pressure $p \in Q$ of an incompressible fluid with negligible advective forces by solving the following variational formulation: find (u, p) such that

$$\begin{aligned} \mu \int_\Omega \nabla u : \nabla v \, dx \; - \; \int_\Omega \operatorname{div} v \, p \, dx &= \int_\Omega f \, v \, dx \quad && \forall v \in V_0, \\ - \int_\Omega \operatorname{div} u \, q \, dx &= 0 && \forall q \in L^2(\Omega), \end{aligned}$$

with $V_0 \subset (H^1(\Omega))^d$, dynamic viscosity μ, and $\partial\Omega_D = \partial\Omega$.

In our numerical tests, we consider the (leaky) lid-driven cavity Stokes problem (cf. [30, sect. 3.1]), which we will denote as the *LDC Stokes problem*: let Ω be the unit square or cube in two or three dimensions, respectively. We prescribe the velocity boundary conditions by

$$\begin{aligned} &d = 2 : \; g = (1, 0)^T \;\text{ if }\; x_2 = 1, \; g = (0, 0)^T \;\text{ if }\; x_2 < 1, \\ &d = 3 : \; g = (1, 0, 0)^T \text{ if } x_3 = 1, \; g = (0, 0, 0)^T \text{ if } x_3 < 1, \end{aligned}$$

where $Q = L^2_0(\Omega)$, and we choose $\mu = 1$ and $f \equiv 0$ for the Stokes problems. Further, we consider a problem which we will denote as the *channel Stokes problem*: let Ω be a channel with $\Omega = [0, 4] \times [0, 1]$ and $\Omega = [0, 4] \times [0, 1]^2$ in two and three dimensions,

respectively. The inflow and outflow boundary conditions are

$$\begin{aligned} &d = 2: \ g = (4x_2(1-x_2), 0)^T && \text{on } \partial\Omega_{D_{\text{in}}}, \\ &d = 3: \ g = (16x_2(1-x_2)x_3(1-x_3), 0, 0)^T && \text{on } \partial\Omega_{D_{\text{in}}}, \\ &\frac{\partial u}{\partial n} - pn = 0 && \text{on } \partial\Omega_N, \end{aligned}$$

with the outward pointing normal vector n and Neumann boundary $\partial\Omega_N$. In two and three dimensions the Neumann boundary is $\partial\Omega_N = \{(4, x_2)^T \in \mathbb{R}^2 : 0 < x_2 < 1\}$ and $\partial\Omega_N = \{(4, x_2, x_3) \in \mathbb{R}^3 : 0 < x_2, x_3 < 1\}$, respectively. On the remainder of the boundary, we set no-slip boundary conditions and $Q = L^2(\Omega)$.

2.2. Navier–Stokes equations. Second, we consider the steady-state Navier–Stokes equations modeling the flow of an incompressible Newtonian fluid with kinematic viscosity $\nu > 0$. We seek to determine the velocity $u \in V_g$ and the pressure $p \in Q$ by solving the following variational formulation: find (u, p) such that

$$\begin{aligned} &\nu \int_\Omega \nabla u : \nabla v \, dx + \int_\Omega (u \cdot \nabla u) \cdot v \, dx - \int_\Omega \operatorname{div} v \, p \, dx = \int_\Omega f \, v \, dx \quad \forall v \in V_0, \\ &- \int_\Omega \operatorname{div} u \, q \, dx = 0 \qquad \forall q \in L^2(\Omega). \end{aligned}$$

The presence of the convection term $u \cdot \nabla u$ leads to a nonlinear system that is solved by Newton's method or Picard iterations; cf. [30, sect. 8.3].

We will consider two different Navier–Stokes model problems for the numerical experiments. In both cases, the source function is $f \equiv 0$.

The first model problem is a regularized lid-driven cavity Navier–Stokes problem, similar to the two-dimensional problem in [30, sect. 3.1], with $\partial\Omega_D = \partial\Omega$ and $Q = L^2_0(\Omega)$. We refer to it as the *LDC Navier–Stokes problem*. The boundary values of the problem are given by

$$\begin{aligned} &d = 2: \ g = (4x_1(1-x_1), 0)^T \text{ if } x_2 = 1, \ g = (0,0)^T \text{ if } x_2 < 1, \\ &d = 3: \ g = (16x_1(1-x_1)x_2(1-x_2), 0, 0)^T \text{ if } x_3 = 1, \ g = (0,0,0)^T \text{ if } x_3 < 1. \end{aligned}$$

The second model problem is the three-dimensional Navier–Stokes benchmark for the flow around a cylinder with circular cross-section, which we will refer to as the *Navier–Stokes benchmark*; cf. [61], where a detailed description of the simulation setup is given. See Figure 1 for the benchmark geometry and the velocity of the solution of the Navier–Stokes benchmark. The length of the domain is $2.5m$, and $A = 0.41m$ is the height and width of the domain. Further, $\nu = 10^{-3}m/s$, and we define $V_g = \{v \in H^1(\Omega)^d : v_{|\partial\Omega_D} = g\}$, $V_0 = \{v \in H^1(\Omega)^d : v_{|\partial\Omega_D} = 0\}$, and $Q = L^2(\Omega)$. The inflow and outflow boundary conditions are

$$\begin{aligned} u &= (16u_{max}x_2x_3(A-x_2)(A-x_3)/A^4, 0, 0)^T \\ \text{on } \partial\Omega_{D_{\text{in}}} &= \{(0, x_2, x_3) \in \mathbb{R}^3 : 0 < x_2, x_3 < A\} \text{ and} \\ \nu\frac{\partial u}{\partial n} - pn = 0 \text{ on } \partial\Omega_N &= \{(2.5, x_2, x_3) \in \mathbb{R}^3 : 0 < x_2, x_3 < A\}, \end{aligned}$$

respectively. Then, the maximum velocity across the inflow is $u_{max} = 0.45m/s$. On the remainder of the boundary, we set no-slip boundary conditions.

In a dimensionless reformulation of the Navier–Stokes equations, the Reynolds number Re specifies the relative contributions of convection and diffusion. For the LDC Navier–Stokes problem in two and three dimensions, we obtain $Re = 1/\nu$. The Reynolds number of the benchmark problem is $Re = 20$.

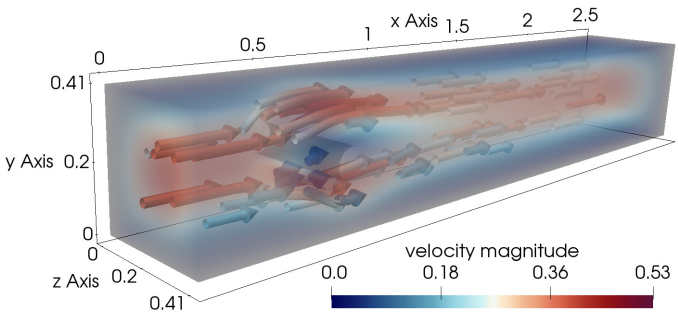

FIG. 1. *Velocity of the Navier–Stokes benchmark problem; Re* $= 20$, $(x, y, z) = (x_1, x_2, x_3)$.

3. Finite element discretization. For the discretization of the saddle point problems considered here, we use mixed finite elements methods. Therefore, we first introduce a triangulation τ_h of Ω with characteristic mesh size h, which can be nonuniform. Then, we introduce the conforming discrete piecewise quadratic velocity and piecewise linear pressure spaces

$$V^h(\Omega) = \{v_h \in (\mathcal{C}(\bar{\Omega}))^d \cap V : v_h|_T \in P_2 \,\forall\, T \in \tau_h\} \text{ and}$$
$$Q^h(\Omega) = \{q_h \in \mathcal{C}(\bar{\Omega}) \cap Q : q_h|_T \in P_1 \,\forall\, T \in \tau_h\},$$

respectively, of P2-P1 Taylor–Hood mixed finite elements. Further, we use piecewise linear velocity and piecewise linear pressure spaces, i.e., P1-P1. The corresponding velocity space is

$$V^h(\Omega) = \{v_h \in (C(\bar{\Omega}))^d \cap V : v_h|_T \in P_1 \,\forall\, T \in \tau_h\},$$

and the corresponding pressure space is the same as for P2-P1. Note that the combination of P1-P1 finite elements does not satisfy the inf-sup condition; cf., e.g., [13]. Therefore, we use the stabilization presented in [30, sect. 3.3]. Here, $\mathcal{C}(\bar{\Omega})$ denotes the space of continuous functions on $\bar{\Omega}$. From now on, we denote by u and p the nodal values of the finite element functions u_h and p_h.

The resulting discrete saddle point problem has the form

$$\mathcal{A}x = \begin{bmatrix} A & B^T \\ B & -C \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix} = \begin{bmatrix} F \\ 0 \end{bmatrix} = b, \tag{1}$$

where $C \neq 0$ is a positive semidefinite matrix, which arises from the stabilization of P1-P1 finite elements and $C = 0$ if no stabilization is used. Note that, in contrast to finite element discretizations with discontinuous pressure, the pressure cannot be eliminated by static condensation on the element level for continuous pressure.

For the model problems considered here, the matrix B^T arises from the discretization of the divergence term

$$\int_\Omega \operatorname{div} v_h \, p_h \, dx.$$

Therefore, the null space of B^T consists of all constant pressure functions. If the matrix C is symmetric positive definite, the pressure is uniquely determined. However, for problems with $\partial\Omega_D = \partial\Omega$, it follows from the divergence theorem that the pressure has zero mean value.

Therefore, we restrict the pressure to the space

$$\overline{Q}^h = Q^h \cap L_0^2(\Omega),$$

i.e., to zero mean value. In order to do so, we can introduce a Lagrange multiplier λ to enforce

$$\int_\Omega p_h \, dx = 0. \tag{2}$$

This results in the block system

$$\overline{\mathcal{A}}\overline{x} = \begin{bmatrix} A & B^T & 0 \\ B & -C & a^T \\ 0 & a & 0 \end{bmatrix} \begin{bmatrix} u \\ p \\ \lambda \end{bmatrix} = \begin{bmatrix} F \\ 0 \\ 0 \end{bmatrix}, \tag{3}$$

where the vector a arises in the finite element discretization of the integral (2). Alternatively, we can use the projection

$$\overline{P} = I_p - a^T(aa^T)^{-1}a \tag{4}$$

onto the space $\overline{Q}^h$, where I_p is the pressure identity matrix.

In our derivations of the new monolithic preconditioners, we concentrate on fluid flow problems where $\partial\Omega_D = \partial\Omega$, such that the pressure is normalized by (2). In section 6, we will give a brief overview of other approaches to enforcing a zero mean value for the pressure. In the case of a global problem with natural boundary conditions with $\partial\Omega_N \neq \emptyset$, the pressure does not belong to $\overline{Q}^h$, and we use a saddle point system formulation as in (1). In section 8, we will present numerical results for both types of problems.

We solve the discrete saddle point problem iteratively using a Krylov subspace method. Since the system becomes very ill-conditioned for small h, we need a scalable preconditioner to guarantee fast convergence of the iterative method. Therefore, we will use monolithic two-level Schwarz preconditioners; cf. section 5.

4. Two-level overlapping Schwarz methods for elliptic problems. Before we define monolithic overlapping Schwarz preconditioners for saddle point problems, we recall their definition for elliptic model problems. Therefore, we consider the linear equation system

$$Ax = b \tag{5}$$

arising from a finite element discretization of an elliptic problem, e.g., a Poisson or elasticity problem on Ω with sufficient Dirichlet boundary conditions.

Let Ω be decomposed into nonoverlapping subdomains $\{\Omega_i\}_{i=1}^N$ with typical diameter H and corresponding overlapping subdomains $\{\Omega_i'\}_{i=1}^N$ with k layers of overlap, i.e., $\delta = kh$; cf. Figure 2 for a visualization of a cubic sample domain. The overlapping subdomains can be constructed from the nonoverlapping subdomains by recursively adding one layer of elements after another to the subdomains. Even if no geometric information is given, this can be performed based on the graph of the stiffness matrix A. Furthermore, let

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

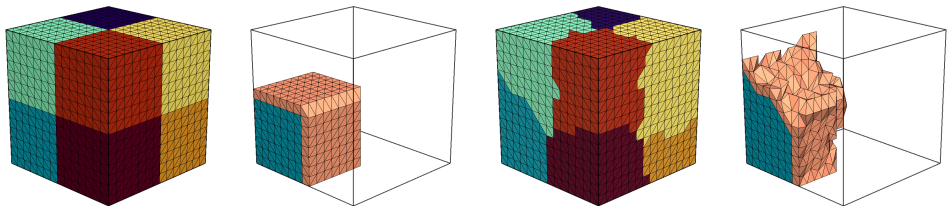

FIG. 2. *Structured mesh and domain decomposition (first and second images) and unstructured mesh and domain decomposition (third and fourth images) of a unit cube: Different colors for each subdomain (first and third images) and one subdomain with one highlighted layer of overlap (second and fourth images). The unstructured mesh is generated as presented in Figure 6, and the unstructured domain decomposition is performed using* `ParMETIS` [46].

be the interface of the nonoverlapping domain decomposition.

We define $R_i : V^h \to V_i^h$, $i = 1, \dots, N$, as the restriction from the global finite element space $V^h = V^h(\Omega)$ to the local finite element space $V_i^h := V^h(\Omega_i')$ on the overlapping subdomain Ω_i'; R_i^T is the corresponding prolongation from V_i^h to V^h. In addition, let V_0 be a global coarse space and $R_0^T : V_0^h \to V$ the corresponding coarse interpolation.

For the time being, we will use exact local solvers; in section 8, we will also present results for inexact coarse solvers. For exact local and coarse solvers, the additive Schwarz preconditioner in matrix form can be written as

$$B_{\mathrm{OS}-2}^{-1} = \underbrace{R_0^T A_0^{-1} R_0}_{\text{coarse level}} + \underbrace{\sum_{i=1}^{N} R_i^T A_i^{-1} R_i}_{\text{first level}}, \tag{6}$$

where the local and coarse stiffness matrices A_i are given by

$$A_i = R_i A R_i^T, \text{ for } i = 0, \dots, N.$$

One typical choice for the coarse basis functions is Lagrangian basis functions on a coarse triangulation. However, a coarse triangulation may not be available for arbitrary geometries and domain decompositions. Therefore, we consider GDSW coarse spaces, which do not require a coarse triangulation.

4.1. The GDSW coarse space. The GDSW preconditioner, which was introduced by Dohrmann, Klawonn, and Widlund in [19, 20], is a two-level additive overlapping Schwarz preconditioner with energy-minimizing coarse space and exact solvers. Thus, the preconditioner can be written in the form

$$B_{\mathrm{GDSW}}^{-1} = \Phi A_0^{-1} \Phi^T + \sum_{i=1}^{N} R_i^T A_i^{-1} R_i, \tag{7}$$

where A_i and R_i are defined as before and

$$A_0 = R_0 A R_0^T = \Phi^T A \Phi$$

is the matrix of the coarse problem. This corresponds to (6), whereas we replace R_0^T by Φ, as is standard in the context of GDSW preconditioners.

For the GDSW preconditioner, the choice of the matrix Φ is the main ingredient. In order to define the columns of Φ, i.e., the coarse basis functions, a partition of unity of energy-minimizing functions and the null space of the operator A are employed.

In particular, the interface Γ is divided into M connected components Γ_j, which are common to the same set of subdomains, i.e., into vertices, edges, and, in three dimensions, faces. Now, let Z be the null space of the global Neumann matrix and Z_{Γ_j} the restriction of Z to the degrees of freedom corresponding to the interface component Γ_j. Then, we construct corresponding matrices Φ_{Γ_j}, such that their columns form a basis of the space Z_{Γ_j}.

Let $R_{\Gamma j}$ be the restriction from Γ onto Γ_j; then the values of the GDSW basis functions Φ on Γ can be written as

$$\Phi_\Gamma = \begin{bmatrix} R_{\Gamma_1}^T \Phi_{\Gamma_1} & \dots & R_{\Gamma_M}^T \Phi_{\Gamma_M} \end{bmatrix}.$$

In particular, after partitioning the degrees of freedom into interface (Γ) and interior (I) ones, the matrix A can be written as

$$A = \begin{bmatrix} A_{II} & A_{I\Gamma} \\ A_{\Gamma I} & A_{\Gamma\Gamma} \end{bmatrix}.$$

Then, the basis functions of the GDSW coarse space can be written as discrete harmonic extensions of Φ_Γ into the interior degrees of freedom:

$$\Phi = \begin{bmatrix} \Phi_I \\ \Phi_\Gamma \end{bmatrix} = \begin{bmatrix} -A_{II}^{-1} A_{I\Gamma} \Phi_\Gamma \\ \Phi_\Gamma \end{bmatrix}. \tag{8}$$

Note that $A_{II} = \operatorname{diag}_{i=1}^N (A_{II}^{(i)})$ is a block diagonal matrix containing the local matrices $A_{II}^{(i)}$ from the nonoverlapping subdomains. Its factorization can thus be computed block by block and in parallel.

The condition number estimate for the GDSW preconditioner

$$\kappa \left(B_{\mathrm{GDSW}}^{-1} A \right) \leq C \left(1 + \frac{H}{\delta} \right) \left(1 + \log \left(\frac{H}{h} \right) \right)^2$$

holds for polygonal/polyhedral Lipschitz domains and certain assumptions on the subdomains. In two dimensions it also holds for the general case of Ω decomposed into John domains; cf. [19, 20]. For certain variants of the GDSW coarse space, the factor $\left(1 + \log\left(\frac{H}{h}\right)\right)^2$ can be improved to $\left(1 + \log\left(\frac{H}{h}\right)\right)$; see [21, 23].

5. Monolithic two-level overlapping Schwarz preconditioners for saddle point problems. Monolithic two-level Schwarz preconditioners for saddle point problems are characterized by the fact that the local problems and the coarse problems have the same block structure as the global saddle point problem.

In contrast, block preconditioners typically omit some of the blocks. Consequently, the convergence of monolithic preconditioners is typically significantly faster than that of block preconditioners; cf., e.g., [51, 34].

5.1. Schwarz preconditioners with Lagrangian coarse spaces. We first introduce a monolithic two-level overlapping Schwarz preconditioner, where the coarse basis consists of Taylor–Hood or P1-P1 finite elements. This approach was introduced by Klawonn and Pavarino in [50, 51] for monolithic preconditioners. We decompose

the spaces V^h and Q^h into local spaces

$$V_i^h = V^h \cap (H_0^1(\Omega_i'))^d \text{ and}$$
$$Q_i^h = Q^h \cap L^2(\Omega_i'),$$

$i = 1, \ldots, N$, respectively, defined on the overlapping subdomains Ω_i'.

We define restriction operators

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

to overlapping subdomains Ω_i', $i = 1, \ldots, N$, for velocity and pressure degrees of freedom, respectively. Consequently, $R_{i,u}^T$ and $R_{i,p}^T$ are prolongation operators from local spaces to the global spaces. The monolithic restriction operators

$$\mathcal{R}_i : V^h \times Q^h \longrightarrow V_i^h \times Q_i^h,$$

$i = 1, \ldots, N$, have the form

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

In addition to that, we introduce local projections

$$\overline{\mathcal{P}}_i : V_i^h \times Q_i^h \longrightarrow V_i^h \times \overline{Q}_i^h,$$

where $\overline{Q}_i^h$ is the local pressure space with zero mean value:

$$\overline{Q}_i^h = \{q_h \in Q^h \cap L_0^2(\Omega_i') : \operatorname{supp}(q_h) \subset \Omega_i'\} \subset Q_i^h.$$

Similar to the global projection (4), we define local projections

$$\overline{P}_i = I_{p,i} - a_i^T (a_i a_i^T)^{-1} a_i, \quad i = 1, \ldots, N, \tag{9}$$

where $I_{p,i}$ is the local pressure identity operator and a_i corresponds to the discretizations of (2) on subdomain Ω_i'. Further, the local velocity identity operators are $I_{u,i}$. Then, we can enforce zero mean value on the local problems using the projection

$$\overline{\mathcal{P}}_i := \begin{bmatrix} I_{u,i} & 0 \\ 0 & \overline{P}_i \end{bmatrix}.$$

The monolithic two-level overlapping Schwarz preconditioner with Lagrangian coarse space reads

$$\hat{\mathcal{B}}_{\mathrm{P2-P1}}^{-1} = \mathcal{R}_0^T \mathcal{A}_0^+ \mathcal{R}_0 + \sum_{i=1}^{N} \mathcal{R}_i^T \overline{\mathcal{P}}_i \mathcal{A}_i^{-1} \mathcal{R}_i, \tag{10}$$

where $\mathcal{A}_0^+$ is a pseudoinverse of the coarse matrix $\mathcal{A}_0 = \mathcal{R}_0 \mathcal{A}_0 \mathcal{R}_0^T$. Since the pressure may not be uniquely determined by the coarse problem, in general, we have to use a pseudoinverse. In practice, we restrict the coarse pressure to $\overline{Q}^H$ for the solution of the coarse problem. Otherwise, if $\mathcal{A}$ is nonsingular, $\mathcal{A}_0^+ = \mathcal{A}_0^{-1}$. Also note that,

due to the projections $\overline{\mathcal{P}}_i$, the preconditioner $\hat{\mathcal{B}}_{\mathrm{P2-P1}}$ is not symmetric. The local matrices

$$\mathcal{A}_i = \mathcal{R}_i \mathcal{A} \mathcal{R}_i^T, \quad i = 1, \ldots, N, \tag{11}$$

are extracted from the global matrix $\mathcal{A}$ and have homogeneous Dirichlet boundary conditions for both velocity and pressure. As we will describe in section 6, there are several ways to treat the pressure in the local overlapping problems. Here, we ensure zero mean value for the pressure of the local contributions using projections $\overline{\mathcal{P}}_i$ in addition to the Dirichlet boundary data in the pressure.

Note that the local overlapping problems and the coarse problem are saddle point problems of the structure (1). Similar to the local problems, the coarse problem with Lagrangian basis functions is defined on a corresponding product space $V_0^h \times Q_0^h$, resulting in the coarse interpolation matrix

$$\mathcal{R}_0^T = \begin{bmatrix} R_{0,u}^T & 0 \\ 0 & R_{0,p}^T \end{bmatrix}, \tag{12}$$

which contains the interpolations of the corresponding coarse basis functions. Therefore, a coarse triangulation is needed for the construction of the coarse level. This is problematic for arbitrary geometries and, in particular, cannot be performed in an algebraic fashion.

Since the coarse pressure has zero mean value due to the specific pseudoinverse $\mathcal{A}_0$, we can omit a projection of the coarse solution onto $\overline{Q}^H$.

5.2. Schwarz preconditioners with GDSW coarse spaces. The new monolithic GDSW preconditioner for saddle point problems is a two-level Schwarz preconditioner with discrete saddle point harmonic coarse space. Therefore, it can be seen as an extension of GDSW for elliptic problems, as described in subsection 4.1, to saddle point problems. Its first level is defined as in subsection 5.1, and therefore, the preconditioner can be written as

$$\hat{\mathcal{B}}_{\mathrm{GDSW}}^{-1} = \phi \mathcal{A}_0^+ \phi^T + \sum_{i=1}^{N} \mathcal{R}_i^T \overline{\mathcal{P}}_i \mathcal{A}_i^{-1} \mathcal{R}_i. \tag{13}$$

The coarse operator reads

$$\mathcal{A}_0 = \phi^T \mathcal{A} \phi, \tag{14}$$

with the coarse basis functions being the columns of the matrix ϕ. The coarse space is constructed in a similar way as in subsection 4.1. As for elliptic problems, the problem is first partitioned into interface (Γ) and interior (I) degrees of freedom. Correspondingly, the matrix $\mathcal{A}$ can be written as

$$\mathcal{A} = \begin{bmatrix} \mathcal{A}_{II} & \mathcal{A}_{I\Gamma} \\ \mathcal{A}_{\Gamma I} & \mathcal{A}_{\Gamma\Gamma} \end{bmatrix}.$$

Each of the submatrices $\mathcal{A}_{**}$ is a block matrix of the form (1).

The coarse basis functions are then constructed as discrete saddle point harmonic extensions of the interface values ϕ_Γ, i.e., as the solution of the linear equation system

$$\begin{bmatrix} \mathcal{A}_{II} & \mathcal{A}_{I\Gamma} \\ 0 & I \end{bmatrix} \begin{bmatrix} \phi_I \\ \phi_\Gamma \end{bmatrix} = \begin{bmatrix} 0 \\ \phi_\Gamma \end{bmatrix}. \tag{15}$$

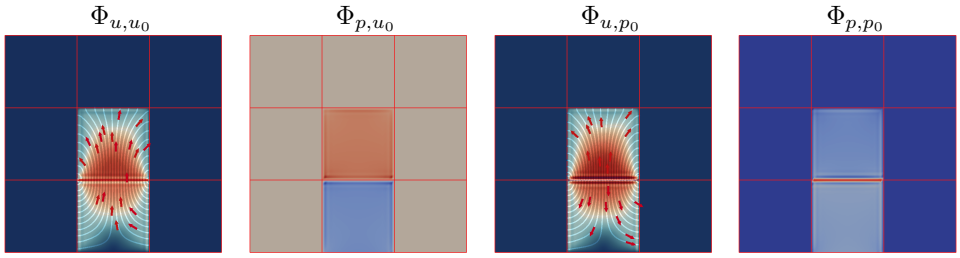

FIG. 3. *Velocity Φ_{u,u_0} (first image) and pressure Φ_{p,u_0} (second image) components of a velocity edge basis function in y-direction. Velocity Φ_{u,p_0} (third image) and pressure Φ_{p,p_0} (fourth image) components of the pressure basis function corresponding to the same edge; see* (16) *for the block structure of Φ. Saddle point harmonic extension for the Stokes equations in two dimensions with* 9 *subdomains.*

The interface values of the coarse basis

$$\phi_\Gamma = \begin{bmatrix} \Phi_{\Gamma,u_0} & 0 \\ 0 & \Phi_{\Gamma,p_0} \end{bmatrix}$$

are decomposed into velocity (u_0) and pressure (p_0) based basis functions.

In particular, the columns of Φ_{Γ,u_0} and Φ_{Γ,p_0} are the restrictions of the null spaces of the operators A and B^T to the interface components Γ_j, $j = 1,\dots,M$; cf. subsection 4.1. Typically, the null space of the operator B^T consists of all pressure functions that are constant on Ω; cf. section 3. Therefore, the columns of Φ_{Γ,p_0} are chosen to be the restrictions of the constant function 1 to the faces, edges, and vertices.

Note that, in contrast to the Lagrangian coarse basis functions (12), where the basis functions do not couple the velocity and pressure degrees of freedom, the off-diagonal blocks in the block representation

$$\phi = \begin{bmatrix} \Phi_{u,u_0} & \Phi_{u,p_0} \\ \Phi_{p,u_0} & \Phi_{p,p_0} \end{bmatrix} \tag{16}$$

are, in general, not zero for the discrete saddle point harmonic coarse spaces; cf. Figure 3. This is beneficial for the scalability of the method if pressure normalization is enforced by condition (2): if the blocks Φ_{u,p_0} and Φ_{p,u_0} are omitted, numerical scalability deteriorates.

If, on the other hand, the pressure is fixed through a boundary condition, the off-diagonal blocks have to be omitted instead to obtain good numerical scalability. Then, ϕ reads

$$\phi = \begin{bmatrix} \Phi_{u,u_0} & 0 \\ 0 & \Phi_{p,p_0} \end{bmatrix}. \tag{17}$$

This also reduces the computational cost of the Galerkin product (14).

Construction of ϕ_Γ for our model problems. For Stokes and Navier–Stokes problems in two dimensions, each interface node is set to

$$r_{u,1} := \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad r_{u,2} := \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad r_{p,1} := \begin{bmatrix} 1 \end{bmatrix}. \tag{18}$$

In three dimensions, each interface node is set to

$$r_{u,1} := \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \quad r_{u,2} := \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \quad r_{u,3} := \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \quad r_{p,1} := \begin{bmatrix} 1 \end{bmatrix}. \tag{19}$$

TABLE 1

LDC Stokes problem in two dimensions using $H/h = 8$ and Taylor–Hood elements; GMRES iteration counts for both versions of local problems for a one-level Schwarz preconditioner without coarse level. Stopping criterion is a residual reduction of 10^{-6}. $*_D$ and $*_N$ indicate Dirichlet and Neumann boundary of the local problems, respectively, for the velocity u or the pressure p.

N	4	9	16	25	36
$u_D p_D$, $\delta = 1h$	23	39	62	83	101
$u_D p_N$, $\delta = 1h$	18	28	44	61	79
$u_D p_D$, $\delta = 2h$	16	26	37	52	64
$u_D p_N$, $\delta = 2h$	12	24	35	44	57

Additionally, we add one basis function for the Lagrange multiplier as will be explained in section 6 for the LDC Stokes and the LDC Navier–Stokes problems; cf. (22).

6. Treating the pressure. As already mentioned in section 3, there are several ways to make the pressure unique. In particular, imposing Dirichlet boundary conditions for the pressure or restricting the pressure to the space $L_0^2(\Omega)$ is a possible way. As pointed out in subsection 5.1, the local overlapping problems possess homogeneous Dirichlet boundary conditions for the pressure. Additionally, we impose zero mean value by projecting the local pressure onto the subspace $L_0^2(\Omega_i)$, $i = 1, \dots, N$.

Another way of imposing zero mean value locally is to introduce local Lagrange multipliers, resulting in local matrices

$$\overline{\mathcal{A}}_i = \begin{bmatrix} A_i & B_i^T & 0 \\ B_i & -C_i & a_i^T \\ 0 & a_i & 0 \end{bmatrix}, \quad i = 1, \dots, N, \tag{20}$$

where the vector a_i arises in the finite element discretization of the integral $\int_{\Omega_i'} p_h \, dx$. In this case, we omit the projections $\overline{\mathcal{P}}_i$ in (10) or (13).

On the other hand, the matrix $\overline{\mathcal{A}}_i$ remains nonsingular even if, instead of Dirichlet boundary conditions, Neumann boundary conditions are imposed for the local pressure. Comparing both approaches separately for a monolithic one-level Schwarz preconditioner, we can observe that imposing Neumann boundary conditions performs slightly better; cf. Table 1. However, this approach is not suitable for an algebraic implementation since, in general, we do not have access to the local Neumann matrices. Therefore, we use the variant with Dirichlet boundary conditions.

Further, for the construction of the monolithic Schwarz preconditioners described in section 5, the local projections $\overline{P}_i$ have to be built. In practice, we do not compute the local projection matrices (9) explicitly but just implement the application of $\overline{P}_i$, $i = 1, \dots, N$, to a vector, requiring access to the local vectors a_i; they could be extracted from the global vector a, which arises in the discretization of (2). If a is not available, geometric information is necessary for the construction of a_i.

For the saddle point problem with Lagrange multiplier (3), we can define an algebraic version of the monolithic GDSW preconditioner. Therefore, we omit the local projections $\overline{\mathcal{P}}_i$ in (13). The resulting two-level preconditioner is then symmetric. In addition, we modify the definition of the restriction operators $\mathcal{R}_i$ such that the Lagrange multiplier is also added to the local problems:

$$\mathcal{R}_i = \begin{bmatrix} R_{i,u} & 0 & 0 \\ 0 & R_{i,p} & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{21}$$

TABLE 2

Iteration counts for the system $\mathcal{A}$ with and without local projections as well as for the system $\overline{\mathcal{A}}$. The latter is the system with global Lagrange multiplier. Two-dimensional LDC Stokes problem using $H/h = 50$, $\delta = 6h$, and Taylor-Hood elements. Iteration counts for the monolithic preconditioner with GDSW coarse space. GMRES is stopped when a reduction of 10^{-6} of the unpreconditioned residual is reached.

N	16	64	196
System $\mathcal{A}$ using local projections	54	57	58
System $\mathcal{A}$ not using local projections	87	196	515
System $\overline{\mathcal{A}}$ with global Lagrange multiplier	56	60	61

Consequently, the local overlapping matrices are of the form (20).

The construction of the GDSW coarse space is also slightly modified. Since the Lagrange multiplier is shared by all subdomains, we treat the Lagrange multiplier as a vertex of the domain decomposition, which is therefore part of the interface Γ. We add one coarse basis function corresponding to the Lagrange multiplier and modify the definition of ϕ_Γ accordingly:

$$\phi_\Gamma = \begin{bmatrix} \Phi_{\Gamma,u_0} & 0 & 0 \\ 0 & \Phi_{\Gamma,p_0} & 0 \\ 0 & 0 & \Phi_{\Gamma,\lambda_0} \end{bmatrix} = \begin{bmatrix} \Phi_{\Gamma,u_0} & 0 & 0 \\ 0 & \Phi_{\Gamma,p_0} & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{22}$$

This monolithic GDSW preconditioner can be built in a purely algebraic fashion from system (3). It reads

$$\hat{\mathcal{B}}^{-1}_{\mathrm{GDSW}} = \phi \mathcal{A}_0^{-1} \phi^T + \sum_{i=1}^{N} \mathcal{R}_i^T \mathcal{A}_i^{-1} \mathcal{R}_i, \tag{23}$$

where $\mathcal{A}_0 = \phi^T \overline{\mathcal{A}} \phi$ is the coarse matrix and the columns of ϕ are the coarse basis functions. They are constructed by extending ϕ_Γ of (22) saddle point harmonically using $\overline{\mathcal{A}}$ from (3) into the interior degrees of freedom; cf. (15). Note that the coarse matrix becomes nonsingular by adding the Lagrange multiplier. Further, the local problems are constructed with restrictions (21), $\mathcal{A}_i = \mathcal{R}_i \overline{\mathcal{A}} \mathcal{R}_i^T$, $i = 1, \ldots, N$.

We observe that both variants, i.e., using local projections and a global Lagrange multiplier, perform equally well; cf. Table 2. Furthermore, without restricting the local pressure to $L^2_0(\Omega)$, the numerical scalability of the two-level monolithic Schwarz preconditioner with GDSW coarse space deteriorates.

7. Implementation details. In this section, we present details on our parallel implementation of the two-level monolithic Schwarz preconditioners for saddle point problems; cf. sections 5 and 6. We refrained from a parallel implementation of the Lagrangian coarse spaces for practical reasons. In particular, the Lagrangian coarse space cannot be constructed in an algebraic fashion for arbitrary geometries; cf. subsection 5.1.

Our parallel implementation is based on the GDSW implementation for elliptic problems in the FROSch framework, which is part of the Trilinos package ShyLU. In particular, we used the earlier `Epetra` version of FROSch which was described in [38] and Trilinos 12.10.0 [43]. First, we will briefly review the GDSW implementation in [38], and then we will discuss its extension to GDSW preconditioners for saddle point problems. We will concentrate on the fully algebraic version described in section 6 for discrete saddle point problems of the form (3). We use one subdomain

per message passing interface (MPI) rank, and, in general, we assume that the global matrix $\mathcal{A}$ is distributed according to the nonoverlapping subdomains of the decomposition; otherwise, we repartition the matrix based on the graph of the matrix using a mesh partitioner in advance.

7.1. GDSW implementation based on Trilinos. The GDSW implementation described in [38] is partitioned into the two classes `SOSSetUp` and `SOS`. The class `SOS` implements

- the computation of the coarse matrix A_0,
- the factorization of the local matrices A_i, $i = 1, \dots, N$, and the coarse matrix A_0, and
- the application of the preconditioner.

The two levels of the Schwarz preconditioner are first applied separately and then summed up. The class `SOSSetUp` sets up the preconditioner; i.e., it

- identifies the overlapping subdomains, builds the local overlapping matrices A_i, $i = 1, \dots, N$, from the global stiffness matrix A, and
- constructs the matrix Φ that contains the coarse basis.

In [38], exact local solvers are used; i.e., the local overlapping matrices A_i, $i = 1, \dots, N$, are extracted from the global stiffness matrix, and direct solvers are used for the solution of the local problems. The coarse basis functions are constructed as described in subsection 4.1. Further, to optimize parallel performance, we drop values from ϕ which are smaller then 10^{-8}.

As the current GDSW implementation in FROSch, our GDSW implementation was restructured compared to [38]. In particular, it is now partitioned into a class for the first level, `AlgebraicOverlappingOperator`, and a class for the coarse level, `GDSWCoarseOperator`. Each of the classes contains both setup and application of the level. Both levels are coupled in an additive way using a `SumOperator`. For more details on the new structure of the implementation, we refer the reader to [37].

For our extension of the GDSW implementation to saddle point problems, we introduce the class `BlockMat`, which implements block operators for `Epetra` objects; see subsection 7.2. This class is used for the implementation of the saddle point problem itself as well as the implementation of the monolithic preconditioner.

7.2. A class for block matrices. To handle the block structure of saddle point problems, we implemented the `BlockMat` class. It is derived from the class `Epetra_Operator`, which defines an abstract interface for arbitrary operators in `Epetra`. To keep the implementation of the `BlockMat` class as general as possible, we allow for various types of blocks, e.g., `Epetra_Operator` or `Epetra_MultiVector` objects. Objects derived from `Epetra_Operator` are, e.g., `Epetra_CrsMatrix` objects, preconditioners like `Ifpack`, `ML`, and the GDSW implementation in [38], or even `BlockMat` objects.

Therefore, the class `BlockMat` could also be used to define block preconditioners, such as block diagonal or block triangular preconditioners, with `Ifpack`, `ML`, or GDSW preconditioners as blocks; cf. [34]. We use the `BlockMat` class throughout the implementation of the monolithic preconditioners to maintain the block structure of all occurring matrices.

7.3. Setup of the first level. As for elliptic problems, in the construction of the monolithic Schwarz preconditioner for saddle point problems, we extract the local overlapping matrices $\overline{\mathcal{A}}_i$ from the global matrix $\overline{\mathcal{A}}$.

In our algebraic implementation, we set up the index sets of the overlapping

```
1 Teuchos::RCP<SOS::SumOperator> additiveSchwarz;
2 std::vector<int> excludeRow(1,3);
3 std::vector<int> excludeCol(1,3);
4 Teuchos::RCP<SOS::OverlappingOperator> firstLevel
5         (new SOS::AlgebraicOverlappingOperator(stokesMatrix,overlap,
              excludeRow,excludeCol));
6 additiveSchwarz->AddOperator(firstLevel);
```

FIG. 4. *Setup of the first level of the monolithic two-level Schwarz preconditioner using the* `BlockMat` *Stokes matrix* `stokesMatrix`*; cf. subsection* 7.2. *The object* `additiveSchwarz` *is a pointer to the* `SumOperator` *which handles the additive coupling of both levels.*

```
1 Teuchos::RCP<SOS::GDSWCoarseOperator> secondLevel(new SOS::
     GDSWCoarseOperator(stokesMatrix));
2 secondLevel->AddBlock(dimension,dofsPerNode,P2MapReplicated,"cont");
3 secondLevel->AddBlock(dimension,P1MapReplicated);
4 secondLevel->AddIdentityBlock();
5 secondLevel->SetUpCoarseGDSWOperator();
6 additiveSchwarz->AddOperator(secondLevel);
```

FIG. 5. *Second level setup of the two-level Schwarz preconditioner using the* `BlockMat` *for the Stokes matrix* `stokesMatrix`*; cf. subsection* 7.2. *The setup of the first level and the* `SumOperator` `additiveSchwarz` *is shown in Figure* 4. *After adding both levels,* `additiveSchwarz` *corresponds to the monolithic GDSW preconditioner.*

subdomains by adding layers of elements recursively based on the graph of the matrix $\mathcal{A}$. In each recursive step, the nonzero pattern of the subdomain matrices has to be gathered on the corresponding MPI ranks, whereas the values of the matrix entries can be neglected. Even if a Lagrange multiplier is introduced to ensure zero mean value of the pressure, we only consider the submatrix $\mathcal{A}$ of $\overline{\mathcal{A}}$ because the Lagrange multiplier couples all pressure degrees of freedom.

Then, the local overlapping submatrices $\overline{\mathcal{A}}_i$, $i = 1, \ldots, N$, are communicated and extracted from $\overline{\mathcal{A}}$ using an `Epetra_Export` object based on the aforementioned local index sets of overlapping subdomains. Then, the matrices are stored in serial `Epetra_CrsMatrix` objects and factorized using direct solvers in serial mode.

A code sample for the setup of the first level is given in Figure 4. The setup is performed by passing the global `BlockMat` $\overline{\mathcal{A}}$ and a user-specified size of overlap. In addition to that, we specify the rows and columns of the block matrix which are excluded from the identification of the overlapping subdomains, i.e., the rows and columns corresponding to the Lagrange multiplier.

7.4. Setup of the coarse level. For the user interface of the coarse level setup, we refer to the lines of code depicted in Figure 5. Each variable, i.e., velocity, pressure, and Lagrange multiplier, is added to the coarse space separately; see lines 2–5 in Figure 5.

As described in section 6, we add one coarse basis function with $\Phi_{\Gamma,\lambda_0} = 1$, which corresponds to the Lagrange multiplier, to the coarse space. Therefore, we use the method `AddIdentityBlock()` in line 4. The velocity and pressure variables are previously added using the `AddBlock()`, specifying the dimension of the domain, the number of degrees of freedom per node, and the ordering of the degrees of freedom in the system. The ordering options are "cont" (continuous), "sep" (separated), and "user" (user-defined). Continuous ordering corresponds to a nodewise ordering,

TABLE 3

LDC Stokes problem in two dimensions with 4 096 subdomains using $H/h = 160$, $\delta = 16h$, and Taylor–Hood elements. Timings for critical parts w.r.t. communication time of the two-level preconditioner with GDSW coarse problem. Standard communication uses `Epetra_Import` and `Epetra_Export` objects only. Modified communication uses additional communication; cf. subsection 7.4. "Avg. apply 1st lvl." and "Avg. apply 2nd lvl." are times averaged over the number of iterations.

Communication	Avg. apply 1st lvl.	Avg. apply 2nd lvl.
Standard	1.42s	0.59s
Modified	1.28s	0.12s

and separated ordering corresponds to dimensionwise ordering, whereas in the user-defined ordering, `Epetra_Map`s have to be provided by the user to define the ordering of the degrees of freedom. `Epetra_Map`s `P2MapReplicated` and `P1MapReplicated` correspond to the velocity and pressure index sets of the nonoverlapping subdomains, respectively. In combination with the ordering, these maps are used to identify the vertices, edges, and, in three dimensions, faces of the domain decomposition; see [38] for a detailed description of this process.

For the computation of the discrete harmonic extensions, we extract the matrices $\overline{\mathcal{A}}_{II}^{(i)}$ and $\overline{\mathcal{A}}_{I\Gamma}^{(i)}$, $i = 1, \dots, N$, from $\overline{\mathcal{A}}$; since we assume that $\overline{\mathcal{A}}$ is distributed according to the nonoverlapping subdomains, no communication is needed for this step. Now, we factorize each $\overline{\mathcal{A}}_{II}^{(i)}$ in serial mode and solve $-\overline{\mathcal{A}}_{II}^{(i)}\phi_I^{(i)} = \overline{\mathcal{A}}_{I\Gamma}^{(i)}\phi_\Gamma^{(i)}$ column by column for $\phi_I^{(i)}$; cf. (15). Then, the global `Blockmat` ϕ is assembled. When `SetUpCoarseGDSWOperator()` is called, the coarse matrix is computed as $\overline{\mathcal{A}}_0 = \phi^T\overline{\mathcal{A}}\phi$, and the resulting globally distributed coarse matrix is reduced to a smaller number of processes using multiple communication steps; cf. [38] for a more detailed discussion. Here, we used three communication steps in all numerical results; in general, this led to the best total performance. Finally, if a direct coarse solver is used, the coarse matrix is factorized in serial or parallel mode.

7.5. Application of the preconditioner. Our global `Epetra_Map` of the solution is uniquely distributed. Therefore, the Lagrange multiplier λ is assigned to only one MPI rank although it belongs to each overlapping subdomain. Also, the coarse basis function corresponding to the Lagrange multiplier couples all subdomains. Consequently, the handling of the Lagrange multiplier in the monolithic preconditioners requires all-to-one and one-to-all communication. As already mentioned in [38], such communication patterns are, in general, not handled well by `Epetra_Export` and `Epetra_Import` objects. One way to overcome this issue is to introduce multiple communication steps. Here, we use `MPI_Broadcast` and `MPI_Reduce` for the communication related to the Lagrange multiplier and `Epetra_Export` and `Epetra_Import` objects for the remaining degrees of freedom. In Table 3, we report the speed-up due to separate communication of the Lagrange multiplier. We save 10% and 80% time in each application of the first and the second level, respectively, for the LDC Stokes problem in two dimensions on 4 096 MPI ranks.

Other than this, the application of the monolithic GDSW preconditioner is handled as for elliptic problems; cf. [38].

8. Numerical results. In this section, we present numerical results of our parallel implementation of the monolithic GDSW preconditioner for the model problems described in section 2. For all model problems which require a pressure normalization (3), we use the preconditioner (23). Only for the Navier–Stokes benchmark and

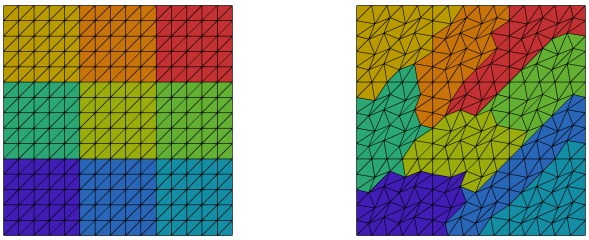

FIG. 6. *Structured (left) and unstructured (right) mesh and decomposition,* $H/h = 5$.

the channel Stokes problem, where no pressure normalization is needed, do we use preconditioner (13) without projections $\overline{\mathcal{P}}_i$. All parallel computations were carried out on the magnitUDE supercomputer at University Duisburg-Essen, Germany. A normal node on magnitUDE has 64GB of RAM and 24 cores (Intel Xeon E5-2650v4 12C 2.2GHz), interconnected with Intel Omni-Path switches. Intel compiler version 17.0.1 and Intel `MKL` 2017 were used.

The two- and three-dimensional Navier–Stokes problems are solved with Newton and Picard iteration, respectively, as we could not observe good convergence with Newton's method without applying globalization techniques in three dimensions. Both methods are started with zero initial guess and the stopping criterion is $\|r_{nl}^{(k)}\|/\|r_{nl}^{(0)}\| \leq 10^{-8}$, with $r_{nl}^{(k)}$ being the kth nonlinear residual.

When using a direct solver for the coarse problem, we employ GMRES [60] for the solution of the linear systems. We use the stopping criterion $\|r^{(k)}\| \leq \varepsilon \|r^{(0)}\|$, where $r^{(k)}$ is the kth unpreconditioned residual and $\varepsilon = 10^{-6}$ is the tolerance for the Stokes problems. For Navier–Stokes problems, we use the tolerance $\varepsilon = 10^{-4}$ for the linear systems. Let us note that, to reduce computational cost, the recurrence relation in GMRES is evaluated until the stopping criterion is satisfied, and only after that, the true residual vector $r^{(k)} = Ax^{(k)} - b$ is computed; this is a standard stopping criterion in the Trilinos package `Belos`. In the case of inexact coarse solves, we employ flexible GMRES (FGMRES) [59] for the outer iterations; cf. subsection 8.1.3. For both Krylov methods, we use the implementations in the Trilinos package `Belos`. As a direct solver, we use `MUMPS` 5.1.1 through the Trilinos `Amesos` interface. The local problems are solved in serial mode, whereas in the case of exact coarse solves, the coarse problems is solved in serial or parallel mode. For inexact coarse solves, we iterate using unpreconditioned GMRES up to a relative tolerance ε_c. The number of MPI ranks for exact and inexact coarse solves is determined by the formula

$$0.5(1 + min\{NumProcs, max\{NumRows/10\,000, NumNonZeros/100\,000\}\}), \tag{24}$$

where $NumRows$ and $NumNonZeros$ are the numbers of rows and nonzeros of $\mathcal{A}_0$, respectively; cf. [38]. In our context, this formula tends to be dominated by the number of nonzeros. If not stated otherwise, we use the above formula to determine the number of MPI ranks used in the coarse solution phase.

Unstructured meshes are constructed from structured meshes by moving the interior nodes; cf. Figure 6. To partition the unstructured meshes in parallel, we use `ParMETIS` 4.0.3 [46]. For the sake of clarity, we characterize all meshes and decompositions by uniform parameters H and h throughout this section.

In the following numerical results, we report first level, second level, and total times. The first level time is the sum of construction and application time for the first

TABLE 4

Iteration counts for an LDC Stokes problem in two dimensions, varying number of subdomains N, and overlap δ with $H/h = 8$ and Taylor–Hood elements. Stopping criterion $\|e^{(k)}\| \leq 10^{-6}$, $e^{(k)} = x^{(k)} - x^*$ with reference solution x^*.

	$\delta = 1h$							$\delta = 2h$						
N	4	9	16	25	36	49	64	4	9	16	25	36	49	64
#its. $\hat{\mathcal{B}}^{-1}_{\text{GDSW}}$	25	33	35	37	38	39	40	21	27	29	32	32	32	33
#its. $\hat{\mathcal{B}}^{-1}_{\text{P2}-\text{P1}}$	21	25	27	28	28	29	29	18	29	21	22	22	22	22

level of the preconditioner, where the application is performed in every outer GMRES or FGMRES iteration. The total time is the sum of both levels.

8.1. Numerical results for Stokes problems. First, we present a comparison of monolithic Schwarz preconditioners with GDSW and standard Lagrangian coarse spaces; cf. subsections 5.1 and 5.2. Further, parallel scalability results for the monolithic GDSW preconditioner are presented for serial and parallel direct coarse solves and inexact coarse solves.

8.1.1. Comparison of Lagrangian and GDSW coarse spaces. Iteration counts for the solution of the LDC Stokes problem in two dimensions using monolithic Schwarz preconditioners with GDSW and Lagrangian coarse spaces are given in Table 4; the results were computed with MATLAB on structured meshes and domain decompositions. Here, GMRES terminates if the error $e^{(k)} = x^{(k)} - x^*$ of the current iterate $x^{(k)}$ to the reference solution x^* of the linear system satisfies $\|e^{(k)}\| \leq 10^{-6}$; as a reference solution we use the one obtained by a direct solver.

We observe that the performance of Lagrangian coarse spaces is slightly better for structured domain decompositions. In contrast, the use of Lagrangian coarse spaces for unstructured decompositions requires additional coarse triangulations and is therefore unfeasible in the context of an algebraic implementation.

Hence, we will focus on the performance of our parallel implementation of the new monolithic GDSW preconditioners for the remainder of section 8.

8.1.2. Parallel scalability using an exact coarse solver. In Table 5, we compare the weak scalability of our monolithic GDSW preconditioner for different levels of overlap δ for the two-dimensional LDC Stokes problem with structured subdomains of constant size $H/h = 160$. We observe very good parallel scalability for every choice of δ; the best efficiency of 77% from 64 to 8 100 MPI ranks is obtained with $\delta = 20h$. Corresponding detailed times are depicted in Figure 7 (left) for up to 8 000 MPI ranks. The time for the first level stays almost constant, whereas a slight increase of the second level time can be observed due to the increasing size of the coarse problem.

In Figure 7 (right), the corresponding times for the three-dimensional LDC Stokes problem with subdomain size $H/h = 10$ are presented. Again, the time for the first level stays constant, whereas a more significant increase of the second level time makes the weak scalability worse than in the two-dimensional case. In particular, the time for the second level exceeds that of the first level when more than 1 000 processor cores are used; see Figure 7 (right). This is typical behavior for two-level methods due to an increased size of the coarse problem and the fact that a direct solver is applied to solve it. A remedy, in order to improve the parallel efficiency, is either to reduce the size of the coarse problem [42] or to introduce a third level [41]. Both approaches, [42] and [41], were successfully applied to second-order elliptic partial

TABLE 5

Iteration counts and weak scalability for two-dimensional LDC Stokes problem with $H/h = 160$, Taylor–Hood elements, and serial coarse solves. Baseline for the efficiency is the fastest time on 64 cores with overlap $\delta = 20h$.

	# cores	64	256	1 024	4 096	8 100
$\delta = 16h$	# its.	66	70	69	68	67
	Time	154.7s	166.4s	172.9s	185.5	202.6s
	Effic.	98%	91%	88%	82%	75%
$\delta = 20h$	# its.	55	56	57	56	56
	Time	151.7s	157.1s	166.9s	180.5	197.2
	Effic.	**100**%	97%	91%	86%	77%
$\delta = 24h$	# its.	50	51	51	51	50
	Time	156.6s	160.9s	169.7s	185.1s	199.4
	Effic.	97%	94%	89%	82%	76%

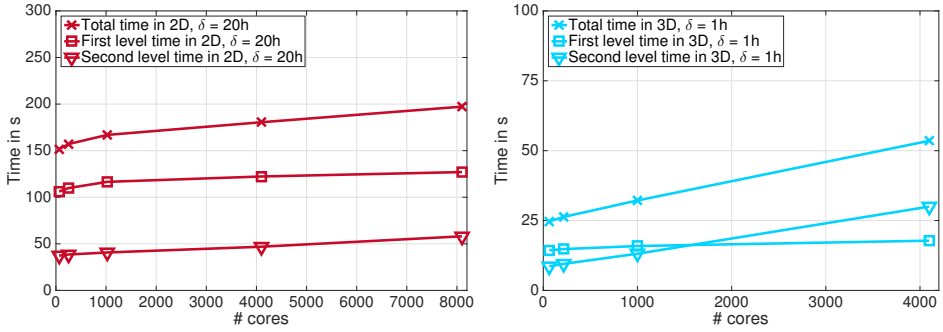

FIG. 7. *First level, second level, and total time for two- and three-dimensional LDC Stokes problems with structured meshes and decompositions using Taylor–Hood elements, 2D $H/h = 160$, 3D $H/h = 10$.*

differential equations. It is a current subject of further investigations to apply them also to the case of saddle point problems, as considered in the present work.

As can be observed from Table 6, the parallel scalability can already be improved slightly by using `Mumps` in parallel mode as the coarse solver; the number of processors used is determined by the formula (24). Consequently, we will always use `Mumps` in parallel mode for the following results with direct coarse solves. Similar results are obtained for the stabilized P1-P1 discretization. In Tables 7 and 8, we present weak scaling results for the two- and three-dimensional channel Stokes problems, respectively.

As can be observed in Table 9 for the three-dimensional LDC Stokes, the dimension of the coarse problem becomes very large for an increasing number of subdomains. In particular, for unstructured domain decompositions, the dimension of the coarse problem and the connectivity of the coarse matrix increase significantly; the dimension can be almost four times, and the number of nonzeros more than ten times, as large as for the structured case.

Therefore, the factorization of the coarse problems with `Mumps` becomes very costly. In subsection 8.1.3, we investigate inexact coarse solves for GDSW coarse problems, where the coarse problems are only solved up to a tolerance ε_c. A similar approach was used in [44] for two-dimensional, stabilized Stokes and Navier–Stokes problems.

TABLE 6

Weak scalability for three-dimensional LDC Stokes problem with a structured mesh and decomposition, $H/h = 10$, and Taylor–Hood elements. Different settings for the coarse problem: `Mumps` used with one MPI rank (serial) and varying number of MPI ranks (parallel) for the coarse problem; "NumProc" denotes the number of MPI ranks used for the computation of the coarse problem determined by (24). Baseline for the efficiency is the fastest time on 64 cores with overlap $\delta = 1h$ and `Mumps` in serial mode.

Overlap	#cores	64	216	1 000	4 096	64	216	1 000	4 096
	NumProc	1	1	1	1	2	9	55	255
$\delta = 1h$	Time	23.5s	26.4s	33.4s	78.4s	24.7s	26.3s	32.2s	53.5s
	Effic.	**100**%	89%	70%	30%	95%	89%	73%	44%
$\delta = 2h$	Time	34.7s	36.9s	44.3s	87.9s	33.1s	35.9s	40.7s	62.5s
	Effic.	67%	64%	53%	27%	71%	65%	58%	38%

TABLE 7

Iteration counts and weak scalability for two-dimensional channel Stokes problem, $H/h = 200$, and stabilized $P1$-$P1$ elements. Baseline for the efficiency is the fastest time on 64 cores with overlap $\delta = 25h$.

	# cores	64	256	1 024	4 096
$\delta = 15h$	# its.	80	83	82	81
	Time	33.0s	36.9s	37.3s	41.5s
	Effic.	94%	84%	83%	75%
$\delta = 20h$	# its.	66	69	70	68
	Time	31.8s	33.5s	35.4s	41.4s
	Effic.	97%	93%	87%	75%
$\delta = 25h$	# its.	56	60	62	60
	Time	31.0s	33.3s	35.3s	38.9s
	Effic.	**100**%	93%	88%	80%

TABLE 8

Iteration counts and weak scalability for three-dimensional channel Stokes problem, $H/h = 20$, and stabilized $P1$-$P1$ elements. Baseline for the efficiency is the fastest time on 64 cores with overlap $\delta = 1h$.

	# cores	108	500	2 048	4 000
$\delta = 1h$	# its.	44	48	48	49
	Time	24.5s	27.5s	32.3s	38.3s
	Effic.	**100**%	89%	76%	64%
$\delta = 2h$	# its.	37	47	47	48
	Time	27.9s	33.1s	36.3s	42.1s
	Effic.	88%	74%	67%	58%

TABLE 9

Number of rows and nonzero entries of the GDSW coarse matrix for the LDC Stokes problem in three dimensions. Interface components of structured and unstructured decompositions.

	#cores	64	216	512	1 000
Structured	NumRows	1 117	4 461	11 453	23 437
	NumNonZeros	165 929	830 409	2 352 361	5 086 985
	Vertices	55	251	687	1 459
	Edges	216	900	2 352	4 860
	Faces	288	1 080	2 688	5 400
Unstructured	NumRows	3 432	15 210	40 757	85 214
	NumNonZeros	1 466 640	8 729 840	26 303 223	58 576 054
	Vertices	389	1 932	5 421	11 587
	Edges	677	2 995	7 887	16 468
	Faces	558	2 254	5 809	11 883

TABLE 10

Weak scalability for the three-dimensional LDC Stokes problem, $\delta = 1h$, and Taylor–Hood elements. The coarse problem is solved exactly with `Mumps` *or with GMRES up to a tolerance ε_c. Baselines for the efficiencies are the fastest times on 64 cores for structured and unstructured meshes and decompositions. Left: Structured mesh and decomposition, $H/h = 10$, using 24 MPI ranks per node. Right: Unstructured mesh and decomposition, $H/h = 13$, using 12 MPI ranks per node.*

Coarse solve	Mesh & partition	Structured				Unstructured			
	#cores	64	216	1 000	4 096	64	216	512	1 000
Exact	Time	22.8s	24.7s	28.1s	49.1s	95.2s	118.9s	135.1s	191.1s
	Effic.	**100**%	92%	81%	46%	98%	78%	69%	49%
	GMRES its.	40	40	38	36	51	52	62	63
$\varepsilon_c = 10^{-1}$	Time	24.3s	26.4s	33.0s	49.4s	95.0s	125.9s	140.1s	143.7s
	Effic.	94%	86%	69%	46%	98%	74%	66%	65%
	FGMRES its.	48	56	75	101	62	68	79	97
$\varepsilon_c = 10^{-2}$	Time	23.0s	25.9s	28.5s	39.4s	93.0s	120.3s	134.6s	149.3s
	Effic.	99%	88%	80%	58%	**100**%	77%	69%	62%
	FGMRES its.	43	53	52	56	54	58	68	79

8.1.3. Parallel scalability using an inexact coarse solver. Instead of solving the coarse problem with a direct method, we solve $\mathcal{A}_0$ iteratively using GMRES, and we use FGMRES for the outer Krylov iterations. In FGMRES, $\hat{\mathcal{B}}$ is used as a right preconditioner, and the kth residual is $r^{(k)} = b - \mathcal{A}\hat{\mathcal{B}}^{-1}(\hat{\mathcal{B}}x^{(k)})$. The coarse problem is solved with GMRES, and $\|r_c^{(k)}\| \leq \varepsilon_c \|r_c^{(0)}\|$, with coarse residual $r_c^{(k)}$, is used as the stopping criterion.

In Table 10, we compare inexact coarse solves using $\varepsilon_c = 10^{-1}, 10^{-2}$ with exact coarse solves for structured and unstructured decompositions. Here, we use right preconditioned GMRES as the iterative Krylov solver in the case of exact coarse solves, such that the residual and the stopping criterion are the same as for FGMRES. We observe that using an inexact coarse solver can be beneficial for both structured and unstructured decompositions. For structured decompositions, inexact coarse solves with $\varepsilon_c = 10^{-2}$ start to be more efficient when using more than 1 000 subdomains, whereas for unstructured decomposition, it is already more efficient for more than 512 cores to use inexact solves.

8.2. Numerical results for Navier–Stokes problems. The scalability results for the LDC Navier–Stokes problems with Reynolds number $Re = 1$ are comparable to the results for the LDC Stokes problems presented in subsection 8.1 and are therefore not reported. In this section, we use $Re = 20$ or $Re = 200$. Inexact coarse solves, as presented in subsection 8.1.3 for the LDC Stokes problem, are not as beneficial for the LDC Navier–Stokes problem. The number of coarse GMRES iterations to reach an acceptable tolerance ε_c is too high, and thus, an additional preconditioner for the coarse problem would be required; cf., e.g., [44]. Therefore, we only consider exact coarse solvers and left preconditioning.

In Table 11 and Table 12, we present weak scalability results for two- and three-dimensional LDC Navier–Stokes problems, respectively. All weak scalability times are averaged over the number of nonlinear Newton or Picard iterations. For $Re = 20$ and a structured decomposition, we observe a slight reduction in iteration counts for finer meshes, as also observed for the LDC Stokes problem; see, e.g., Table 10. Further, the number of Newton or Picard iterations may be reduced when the problem is solved on finer meshes. For two-dimensional problems, we observe 84% parallel efficiency from 64 to 4 096 processor cores, whereas the efficiency deteriorates to 48% in three dimensions; cf. subsection 8.1.2 for a brief discussion on the improvement of computing

TABLE 11

Weak scalability for the two-dimensional LDC Navier–Stokes problem using $H/h = 130$, $\delta = 13h$, and Taylor–Hood elements. Baseline for the efficiencies is the time on 64 cores for each Reynolds number. Times are averages over the number of Newton iterations.

Reynolds number	$Re = 20$				$Re = 200$			
# cores	64	256	1 024	4 096	64	256	1 024	4 096
Time	78.6s	78.0s	86.4s	94.8s	87.4s	84.5s	98.3s	104.0s
Effic.	100%	**102**%	90%	84%	100%	**103**%	89%	84%
Avg. GMRES its.	78.8	77.3	74.5	71.5	101.3	93.3	107.8	99.0
Newton its.	3	3	2	2	4	4	4	4

TABLE 12

Weak scalability for the three-dimensional LDC Navier–Stokes problem using $H/h = 10$, $\delta = 1h$, and Taylor–Hood elements. Baseline for the efficiencies is the time on 64 cores for each Reynolds number. Times are averages over the number of Picard iterations.

Reynolds number	$Re = 20$				$Re = 200$			
# cores	64	216	1 000	4 096	64	216	1 000	4 096
Time	26.0s	27.8s	35.0s	53.7s	28.3s	32.0s	36.7s	59.0s
Effic.	**100**%	93%	74%	48%	**100**%	88%	77%	48%
Avg. GMRES its.	61.8	63.8	63.0	62.3	77.4	86.9	89.9	87.1
Newton/Picard its.	4	4	3	3	18	14	11	7

times for three-dimensional problems. Similar to the results of the LDC Stokes, we observed that an overlap of roughly 10% is most efficient.

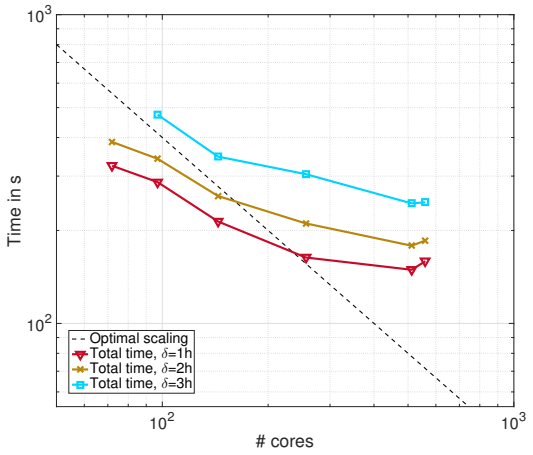

FIG. 8. *Strong scalability for the monolithic GDSW preconditioner applied to the Navier–Stokes benchmark problem with unstructured decompositions and Taylor–Hood elements.*

Finally, we present results for the flow around a cylinder benchmark with circular cross-section and $Re = 20$; see Figure 1 for the velocity field of the solution. We obtain good values for the drag and lift coefficients, $c_D = 6.178$, $c_L = 0.0095$, for a problem with 2.9M degrees of freedom (2.8M velocity, 127K pressure). The domain was partitioned using `METIS`. Both coefficients are computed with the volume integrals given in [45]. Strong scalability results for the monolithic GDSW preconditioner with $\delta = 1h, 2h, 3h$ are presented in Figure 8. The good strong scalability deteriorates from 256 to 512 cores since the additional time to construct and solve the larger

coarse problem exhausts the other time savings.

9. Conclusion. We have presented a new monolithic GDSW preconditioner for saddle point problems and its efficient parallel implementation based on the FROSch framework in Trilinos. The monolithic approach provides robustness and good parallel scalability for up to several thousand cores. Nonetheless, for the problems considered here, the preconditioner can be constructed in an algebraic fashion from the fully assembled saddle point system, and we are therefore able to provide a very reduced and simple user interface to our implementation. Furthermore, unstructured meshes and domain decompositions are no restriction as they are handled in the same way as structured cases. Further improvements of the parallel scalability, in particular for three-dimensional saddle point problems, using reduced coarse spaces (cf. [42]) and multilevel approaches (cf. [41]), are open topics for future research.

REFERENCES

[1] K. J. Arrow, L. Hurwicz, H. Uzawa, and H. B. Chenery, *Studies in Linear and Non-Linear Programming*, Stanford University Press, Stanford, CA, 1958.

[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), e02756.

[3] R. E. Bank, B. D. Welfert, and H. Yserentant, *A class of iterative methods for solving saddle point problems*, Numer. Math., 56 (1990), pp. 645–666.

[4] A. T. Barker and X.-C. Cai, *Scalable parallel methods for monolithic coupling in fluid-structure interaction with application to blood flow modeling*, J. Comput. Phys., 229 (2010), pp. 642–659.

[5] A. T. Barker and X.-C. Cai, *Two-level Newton and hybrid Schwarz preconditioners for fluid-structure interaction*, SIAM J. Sci. Comput., 32 (2010), pp. 2395–2417, https://doi.org/10.1137/090779425.

[6] M. Benzi, G. H. Golub, and J. Liesen, *Numerical solution of saddle point problems*, Acta Numer., 14 (2005), pp. 1–137.

[7] M. Benzi and X.-P. Guo, *A dimensional split preconditioner for Stokes and linearized Navier-Stokes equations*, Appl. Numer. Math., 61 (2011), pp. 66–76.

[8] 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.

[9] D. Braess and C. Blömer, *A multigrid method for a parameter dependent problem in solid mechanics*, Numer. Math., 57 (1990), pp. 747–761.

[10] J. H. Bramble and J. E. Pasciak, *A domain decomposition technique for Stokes problems*, Appl. Numer. Math., 6 (1990), pp. 251–261.

[11] J. H. Bramble, J. E. Pasciak, and A. T. Vassilev, *Analysis of the inexact Uzawa algorithm for saddle point problems*, SIAM J. Numer. Anal., 34 (1997), pp. 1072–1092, https://doi.org/10.1137/S0036142994273343.

[12] S. C. Brenner, *Multigrid methods for parameter dependent problems*, RAIRO Modél. Math. Anal. Numér., 30 (1996), pp. 265–297.

[13] F. Brezzi and M. Fortin, *Mixed and Hybrid Finite Element Methods*, Springer-Verlag, New York, Berlin, Heidelberg, 1991.

[14] X.-C. Cai, D. E. Keyes, and L. Marcinkowski, *Non-linear additive Schwarz preconditioners and application in computational fluid dynamics*, Internat. J. Numer. Methods Fluids, 40 (2002), pp. 1463–1470.

[15] P. Crosetto, S. Deparis, G. Fourestey, and A. Quarteroni, *Parallel algorithms for fluid-structure interaction problems in haemodynamics*, SIAM J. Sci. Comput., 33 (2011), pp. 1598–1622, https://doi.org/10.1137/090772836.

[16] S. Deparis, D. Forti, G. Grandperrin, and A. Quarteroni, *FaCSI: A block parallel preconditioner for fluid–structure interaction in hemodynamics*, J. Comput. Phys., 327 (2016), pp. 700–718.

[17] 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.

[18] C. R. Dohrmann, *Some domain decomposition algorithms for mixed formulations of elasticity and incompressible fluids*, in Workshop on Adaptive Finite Elements and Domain Decomposition Methods, Università degli Studi di Milano, Milan, Italy, 2010.

[19] 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, https://doi.org/10.1137/070685841.

[20] C. R. Dohrmann, A. Klawonn, and O. B. Widlund, *A family of energy minimizing coarse spaces for overlapping Schwarz preconditioners*, in Domain Decomposition Methods in Science and Engineering XVII, Lect. Notes Comput. Sci. Eng. 60, Springer, Berlin, 2008, pp. 247–254.

[21] C. R. Dohrmann and O. B. Widlund, *An overlapping Schwarz algorithm for almost incompressible elasticity*, SIAM J. Numer. Anal., 47 (2009), pp. 2897–2923, https://doi.org/10.1137/080724320.

[22] C. R. Dohrmann and O. B. Widlund, *Hybrid domain decomposition algorithms for compressible and almost incompressible elasticity*, Internat. J. Numer. Methods Engrg., 82 (2010), pp. 157–183.

[23] C. R. Dohrmann and O. B. Widlund, *An alternative coarse space for irregular subdomains and an overlapping Schwarz algorithm for scalar elliptic problems in the plane*, SIAM J. Numer. Anal., 50 (2012), pp. 2522–2537, https://doi.org/10.1137/110853959.

[24] C. R. Dohrmann and O. B. Widlund, *Lower dimensional coarse spaces for domain decomposition*, in Domain Decomposition Methods in Science and Engineering XXI, Springer, Berlin, 2014, pp. 527–535.

[25] C. R. Dohrmann and O. B. Widlund, *On the design of small coarse spaces for domain decomposition algorithms*, SIAM J. Sci. Comput., 39 (2017), pp. A1466–A1488, https://doi.org/10.1137/17M1114272.

[26] 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, https://doi.org/10.1137/040608817.

[27] 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.

[28] H. Elman and D. Silvester, *Fast nonsymmetric iterations and preconditioning for Navier–Stokes equations*, SIAM J. Sci. Comput., 17 (1996), pp. 33–46, https://doi.org/10.1137/0917004.

[29] H. C. Elman and G. H. Golub, *Inexact and preconditioned Uzawa algorithms for saddle point problems*, SIAM J. Numer. Anal., 31 (1994), pp. 1645–1661, https://doi.org/10.1137/0731085.

[30] H. C. Elman, D. J. Silvester, and A. J. Wathen, *Finite Elements and Fast Iterative Solvers: With Applications in Incompressible Fluid Dynamics*, 2nd ed., Numerical Mathematics and Scientific Computation, Oxford University Press, Oxford, UK, 2014.

[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.

[32] M. W. Gee, U. Küttler, and W. A. Wall, *Truly monolithic algebraic multigrid for fluid-structure interaction*, Internat. J. Numer. Methods Engrg., 85 (2011), pp. 987–1016.

[33] A. Heinlein, *Parallel Overlapping Schwarz Preconditioners and Multiscale Discretizations with Applications to Fluid-Structure Interaction and Highly Heterogeneous Problems*, Ph.D. thesis, University of Cologne, Köln, Germany, 2016.

[34] A. Heinlein, C. Hochmuth, and A. Klawonn, *GDSW preconditioners and block-preconditioners for saddle-point problems*, in preparation.

[35] A. Heinlein, A. Klawonn, J. Knepper, and O. Rheinbach, *Adaptive GDSW Coarse Spaces for Overlapping Schwarz Methods in Three Dimensions*, Technical report series, Center for Data and Simulation Science, University of Cologne, Köln, Germany, 2018.

[36] A. Heinlein, A. Klawonn, J. Knepper, and O. Rheinbach, *An adaptive GDSW coarse space for two-level overlapping Schwarz methods in two dimensions*, in Domain Decomposition Methods in Science and Engineering XXIV, Lect. Notes Comput. Sci. Eng. 125, Springer, Cham, 2018, pp. 373–382.

[37] A. Heinlein, A. Klawonn, S. Rajamanickam, and O. Rheinbach, *FROSch—A Parallel Implementation of the GDSW Domain Decomposition Preconditioner in Trilinos*, in preparation.

[38] 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, https://doi.org/10.1137/16M1062843.

[39] A. Heinlein, A. Klawonn, and O. Rheinbach, *Parallel two-level overlapping Schwarz methods in fluid-structure interaction*, in Numerical Mathematics and Advanced Applications—ENUMATH 2015, Lect. Notes Comput. Sci. Eng. 112, Springer, Cham, 2016, pp. 521–530.

[40] A. Heinlein, A. Klawonn, and O. Rheinbach, *Parallel overlapping Schwarz with an energy-minimizing coarse space*, in Domain Decomposition Methods in Science and Engineering XXIII, Lect. Notes Comput. Sci. Eng. 116, Springer, Cham, 2017, pp. 353–360.

[41] A. Heinlein, A. Klawonn, O. Rheinbach, and F. Röver, *A Three-Level Extension of the GDSW Overlapping Schwarz Preconditioner in Two Dimensions*, Tech. report, Technische Universität Bergakademie Freiberg, Freiberg, Germany, 2018; Lect. Notes Comput. Sci. Eng., Springer, Cham, to appear.

[42] A. Heinlein, A. Klawonn, O. Rheinbach, and O. Widlund, *Improving the parallel performance of overlapping Schwarz methods by using a smaller energy minimizing coarse space*, 2017, in Domain Decomposition Methods XXIV, Lect. Notes Comput. Sci. Eng. 125, Springer, Cham, 2018, pp. 383–392.

[43] 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. Softw., 31 (2005), pp. 397–423.

[44] F.-N. Hwang and X.-C. Cai, *Parallel fully coupled Schwarz preconditioners for saddle point problems*, Electron. Trans. Numer. Anal., 22 (2006), pp. 146–162.

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

[46] G. Karypis, K. Schloegel, and V. Kumar, *ParMETIS—Parallel Graph Partitioning and Sparse Matrix Ordering. Version* 3.2, Tech. report, Department of Computer Science and Engineering, University of Minnesota, Twin Cities, Minneapolis, St. Paul, MN, 2011.

[47] 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, https://doi.org/10.1137/S106482759935808X.

[48] A. Klawonn, *Block-triangular preconditioners for saddle point problems with a penalty term*, SIAM J. Sci. Comput., 19 (1998), pp. 172–184, https://doi.org/10.1137/S1064827596303624.

[49] A. Klawonn, *An optimal preconditioner for a class of saddle point problems with a penalty term*, SIAM J. Sci. Comput., 19 (1998), pp. 540–552, https://doi.org/10.1137/S1064827595279575.

[50] A. Klawonn and L. Pavarino, *Overlapping Schwarz methods for mixed linear elasticity and Stokes problems*, Comput. Methods Appl. Mech. Engrg., 165 (1998), pp. 233–245.

[51] A. Klawonn and L. Pavarino, *A comparison of overlapping Schwarz methods and block preconditioners for saddle point problems*, Numer. Linear Algebra Appl., 7 (2000), pp. 1–25.

[52] A. Klawonn and G. Starke, *Block triangular preconditioners for nonsymmetric saddle point problems: Field-of-values analysis*, Numer. Math., 81 (1999), pp. 577–594.

[53] S. Patankar and D. Spalding, *A calculation procedure for heat, mass and momentum transfer in three dimensional parabolic flows*, Internat. J. Heat Mass Transfer, 15 (1972), pp. 1787–1806.

[54] 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, https://doi.org/10.1137/S1064827500372250.

[55] A. Quarteroni, F. Saleri, and A. Veneziani, *Analysis of the Yosida method for the incompressible Navier-Stokes equations*, J. Math. Pures Appl. (9), 78 (1999), pp. 473–503.

[56] 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.

[57] E. Rønquist, *A domain decomposition solver for the incompressible Navier-Stokes equations*, in Workshop on Spectral Element Methods, North Carolina State University, Raleigh, NC, 1994.

[58] T. Rusten and R. Winther, *A preconditioned iterative method for saddlepoint problems*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 887–904, https://doi.org/10.1137/0613054.

[59] Y. Saad, *A flexible inner-outer preconditioned GMRES algorithm*, SIAM J. Sci. Comput., 14 (1993), pp. 461–469, https://doi.org/10.1137/0914028.

[60] Y. Saad and M. H. Schultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Stat. Comput., 7 (1986), pp. 856–869, https://doi.org/10.1137/0907058.

[61] M. Schäfer, S. Turek, F. Durst, E. Krause, and R. Rannacher, *Benchmark computations of laminar flow around a cylinder*, in Flow Simulation with High-Performance Computers II, Notes Numer. Fluid Mech. 48, Vieweg+Teubner Verlag, Wiesbaden, Germany, 1996, pp. 547–566.

[62] D. Silvester, H. Elman, D. Kay, and A. Wathen, *Efficient preconditioning of the linearized Navier-Stokes equations for incompressible flow*, in Numerical Analysis 2000: Partial Differential Equations, Elsevier, New York, 2001, pp. 261–279.

[63] 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.

[64] D. Silvester and A. Wathen, *Fast iterative solution of stabilised Stokes systems part* II. *Using general block preconditioners*, SIAM J. Numer. Anal., 31 (1994), pp. 1352–1367, https://doi.org/10.1137/0731070.

[65] R. S. Tuminaro, C. H. Tong, J. N. Shadid, K. D. Devine, and D. Day, *On a multilevel preconditioning module for unstructured mesh Krylov solvers: Two-level Schwarz*, Commun. Numer. Methods Engrg., 18 (2002), pp. 383–389.

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

[67] R. Verfürth, *A multilevel algorithm for mixed problems*, SIAM J. Numer. Anal., 21 (1984), pp. 264–271, https://doi.org/10.1137/0721019.

[68] A. Wathen and D. Silvester, *Fast iterative solution of stabilised Stokes systems part* I. *Using simple diagonal preconditioners*, SIAM J. Numer. Anal., 30 (1993), pp. 630–649, https://doi.org/10.1137/0730031.

[69] G. Wittum, *Multi-grid methods for Stokes and Navier-Stokes equations*, Numer. Math., 54 (1989), pp. 543–563.

[70] Y. Wu and X.-C. Cai, *A fully implicit domain decomposition based ALE framework for three-dimensional fluid–structure interaction with application in blood flow computation*, J. Comput. Phys., 258 (2014), pp. 524–537.

[71] A. Zsaki, D. Rixen, and M. Paraschivoiu, *A substructure-based iterative inner solver coupled with Uzawa's algorithm for the Stokes problem*, Internat. J. Numer. Methods Fluids, 43 (2003), pp. 215–230.