PCBDDC: A Class of Robust Dual-Primal Methods in PETSc

SIAM J. SCI. COMPUT.
Vol. 38, No. 5, pp. S282–S306

PCBDDC: A CLASS OF ROBUST DUAL-PRIMAL METHODS IN PETSc*

STEFANO ZAMPINI†

Abstract. A class of preconditioners based on balancing domain decomposition by constraints methods is introduced in the Portable, Extensible Toolkit for Scientific Computation (PETSc). The algorithm and the underlying nonoverlapping domain decomposition framework are described with a specific focus on their current implementation in the library. Available user customizations are also presented, together with an experimental interface to the finite element tearing and interconnecting dual-primal methods within PETSc. Large-scale parallel numerical results are provided for the latest version of the code, which is able to tackle symmetric positive definite problems with highly heterogeneous distributions of the coefficients. Current limitations and future extensions of the preconditioner class are also discussed.

Key words. balancing domain decomposition by constraints, BDDC, FETI-DP, PETSc, domain decomposition

AMS subject classifications. 65F08, 65N55, 65Y05, 68W10

DOI. 10.1137/15M1025785

1. Introduction.

The aim of this paper is to present a novel class of preconditioners in the Portable, Extensible Toolbox for Scientific Computation (PETSc) library [10], which are based on the balancing domain decomposition by constraints (BDDC) algorithm [24]. The BDDC methods belong to the family of nonoverlapping domain decomposition methods [79] and they were first developed as primal alternatives of the finite element tearing and interconnecting dual-primal (FETI-DP) [29] methods. BDDC and FETI-DP methods have proven to be powerful preconditioners for finite element discretizations of elliptic partial differential equations (PDEs), with typical polylogarithmic condition numbers bounds of the type

$$\kappa_2 \leq C(1+\log(H/h))^2,$$

with h the characteristic size of the elements, H the maximum diameter of the subdomains, and C a constant independent of the number of subdomains considered [61, 62]. These condition number bounds hold even in the presence of jumps in the coefficient of the PDE between subdomains [61], or when rough interfaces among subdomains are present [50]. A polylogarithmic bound has also been established for some cases with discontinuities crossing subdomain boundaries [70, 71]. Although such bounds are typically obtained under the assumption that each subdomain is the union of a few well-shaped coarse elements of a coarse triangulation, the BDDC and FETI-DP algorithms are well suited for more general and irregular subdomains as given by mesh partitioners.

The BDDC algorithms provide preconditioners for the linear operator of the discretized PDE, whereas the FETI-DP methods recast the linear system as a constrained minimization problem and iterate on the set of Lagrange multipliers introduced as in

*Received by the editors June 15, 2015; accepted for publication (in revised form) April 19, 2016; published electronically October 27, 2016.
http://www.siam.org/journals/sisc/38-5/M102578.html
†Extreme Computing Research Center, Computer, Electrical and Mathematical Sciences and Engineering Division, King Abdullah University of Science and Technology, Thuual 23955, Saudi Arabia (stefano.zampini@kaust.edu.sa).

an associated saddle point formulation. The construction of both methods relies on the selection of *primal continuity constraints* and on the choice of an averaging procedure; the latter is needed in BDDC methods to restore the continuity of the degrees of freedom at the interface between the subdomains during Krylov iterations, whereas in FETI-DP, continuity is restored in the limit by using Lagrange mulipliers. In both cases, the averaging procedure contributes to the robustness of the methods for the case of coefficient jumps between subdomains. For the same choice of the primal constraints and the averaging procedure, the iteration matrices of the two methods have been proven to have the same spectra except for eigenvalues at 0 and 1 [62]. It is worth mentioning that primal variants of FETI-DP were also independently proposed in [23] and [32].

The BDDC method has been successfully extended to different discretization techniques such as the lowest order Nédélec [26] and Raviart-Thomas [65] elements, spectral elements [66], mortar discretizations [38], discontinuous Galerkin [28], and isogeometric analysis [13]. The algorithm also has been applied to a variety of PDEs including advection-diffusion [84], incompressible Stokes [55], porous media flows [82, 83, 73], almost incompressible elasticity [67], Reissner–Mindlin plates [11, 53], and Naghdi shells [12]. The FETI-DP method has also been extensively studied; among others, we cite the studies on the Stokes problem [40], contact problems [6], indefinite complex problems [30, 31], mortar discretizations [39], and biomechanics [49]. Interesting nonlinear formulations of the BDDC and FETI-DP methods have been recently proposed to tackle nonlinear problems with localized nonlinearities [43].

Extensions of the BDDC algorithm have recently focused on primal space enrichments techniques for symmetric positive definite (SPD) systems with high-contrast distributions in the coefficients of the PDE; cf. [15, 19, 41, 42, 44, 45, 75, 69]. All these techniques increase the arithmetic intensity of the algorithm by properly combining generalized eigenvalue calculations and nearest neighbor communications; the outcome is a robust BDDC preconditioner which is able to reduce the number of iterations of the Krylov solver in a black-box fashion. Similar techniques have been studied for enriching the coarse space of overlapping Schwarz [33, 78], FETI [76], and Neumann–Neumann algorithms [77].

The paper is organized as follows. Section 2 describes the nonoverlapping domain decomposition setting and provides details for the construction of specific linear operators with the PETSc library. Section 3 describes the BDDC method and the current PETSc implementation, together with available user customizations of the method; an experimental interface to the FETI-DP method is also introduced. Algorithmic details of the primal space enrichment technique implemented in `PCBDDC` are discussed. Large-scale numerical results obtained with the latest version of the code are provided in section 4. Current limitations and future developments are discussed in section 5.

2. Nonoverlapping domain decomposition. Following the framework of iterative substructuring algorithms [79], the domain Ω is decomposed into N nonoverlapping open Lipschitz subdomains Ω_i such that

$$\overline{\Omega} = \bigcup_{i=1}^{N} \overline{\Omega}_i.$$

Let $\mathbf{a}(\cdot,\cdot)$ be the bilinear form of the variational formulation we are considering and $\mathbf{a}^{(i)}(\cdot,\cdot)$ its restriction to subdomain i, i.e.,

$$\mathbf{a}(\cdot,\cdot) = \int_{\Omega} \dots \ , \quad \mathbf{a}^{(i)}(\cdot,\cdot) = \int_{\Omega_i} \dots \ .$$

We will denote by $\widehat{A}$ the linear operator corresponding to the discretization of $\mathbf{a}(\cdot,\cdot)$ in a finite dimensional Hilbert space $\mathbf{V}_h = \mathbf{V}_h(\Omega)$. Within the nonoverlapping framework, linear operators defined on $\mathbf{V}_h$ are never assembled explicitly. Instead, local operators are assembled at the subdomain level by considering

$$A^{(i)} : \mathbf{V}_h^{(i)} \to \mathbf{V}_h^{(i)'}, \quad \langle A^{(i)} u, v\rangle = \mathbf{a}^{(i)}(u,v), \ \forall\ u, v \in \mathbf{V}_h^{(i)},$$

with $\mathbf{V}_h^{(i)} = \mathbf{V}_h(\Omega_i)$, and $\langle\cdot,\cdot\rangle$ the duality pairing. Let

$$R^{(i)} : \mathbf{V}_h \to \mathbf{V}_h^{(i)}$$

be the restriction operator from the global to the local space; using the property of finite summation of integrals, the action of the global linear operator $\widehat{A}$ can then be expressed by the *subassembling relation*

$$\widehat{A} = R^T A R,$$

where A is a block diagonal matrix with one block for each subdomain, i.e.,

$$A = \begin{bmatrix} A^{(1)} & & \\ & \ddots & \\ & & A^{(N)} \end{bmatrix}, \tag{2.1}$$

and R the direct sum of the $R^{(i)}$ operators.

In order to simplify the notation used for the description of the BDDC algorithm, we write $\mathbf{W}^{(i)} = \mathbf{V}_h^{(i)}$. Within the nonoverlapping framework, the interface Γ between the subdomains is defined as

$$\Gamma = \bigcup_{i \neq j} \partial\Omega_j \cap \partial\Omega_i,$$

and the local discrete spaces are split into interior and interface degrees of freedom

$$\mathbf{W}^{(i)} = \mathbf{W}_I^{(i)} \oplus \mathbf{W}_\Gamma^{(i)}, \tag{2.2}$$

where the degrees of freedom in the interior (I) correspond to those basis functions that are zero on Γ. After the split, the following product spaces are defined:

$$\mathbf{W} = \prod_{i=1}^{N} \mathbf{W}^{(i)}, \quad \mathbf{W}_\Gamma = \prod_{i=1}^{N} \mathbf{W}_\Gamma^{(i)}, \quad \mathbf{W}_I = \prod_{i=1}^{N} \mathbf{W}_I^{(i)}.$$

Since the spaces $\mathbf{W}$ and $\mathbf{W}_\Gamma$ are discontinuous across the interface, their continuous subspaces are usually distinguished by using a surrounding hat,

$$\widehat{\mathbf{W}}_\Gamma \subset \mathbf{W}_\Gamma, \quad \widehat{\mathbf{W}} \subset \mathbf{W}, \quad \widehat{\mathbf{W}} = \mathbf{W}_I \oplus \widehat{\mathbf{W}}_\Gamma. \tag{2.3}$$

The PETSc matrix class for matrices in unassembled format is the `MATIS` type, which has the following constructor:

```
PetscErrorCode  MatCreateIS(MPI_Comm comm, PetscInt bs,
                            PetscInt m,PetscInt n,
                            PetscInt M, PetscInt N,
                            ISLocalToGlobalMapping map, Mat* A)
```

where `comm` is the communicator associated with the matrix and `bs` the block size (i.e., the number of degrees of freedom per grid point). `M` and `N` are the global number of rows and columns of the matrix, whereas `m` and `n` are the local sizes of the distributed vectors used for matrix-vector multiplication and they are not related to the number of unknowns on the subdomains. The current `MATIS` implementation assumes a one-to-one mapping between MPI processes and subdomains, and thus the number of subdomains is given by the size of the communicator. The local to global mapping `map` is the PETSc object representing the action of the operator R^T, and it stores the information needed to subassemble local matrices; the number of unknowns on each subdomain is then inferred by the constructor from the local size of the map.

Figure 1 contains a very simple example of how the `MATIS` object can be constructed. A square (left) is triangulated into two triangles, and the degrees of freedom are labeled by their global number and placed on the vertices of the triangles. In the central panel, each triangle is then assigned to a subdomain (0 or 1): the degrees of freedom are then labeled by the pair (`G`,`L`), where `L` represents the degree of freedom in a local subdomainwise ordering, whereas `G` corresponds to the global ordering. The local to global maps needed to construct the `MATIS` operator are provided on the right, together with the resulting split of local spaces according to (2.2). Note that the current implementation is not limited to P^1 elements, but can accommodate any kind of elements, even with different types, as long as the subdomain matrices and the local to global maps are properly constructed by the user.

Once the `MATIS` object has been constructed, matrix entries can be inserted by using either a local numbering of the degrees of freedom via `MatSetValuesLocal` or a global numbering via `MatSetValues`. Subdomain matrices are stored in the opaque object and they can be accessed using `MatISGetLocalMat`; local subdomain matrices can be replaced using `MatISSetLocalMat`. The function `MatISGetMPIXAIJ` can be used to convert from the `MATIS` format to an assembled `AIJ` object. Matrix preallocation can be locally performed by first retrieving the subdomain matrix from the `MATIS` object and then calling the specific routines for sequential objects; otherwise, if the global distribution of nonzeros is known for the `AIJ` object, the preallocation routine `MatISSetPreallocation` can be used.

3. Balancing domain decomposition by constraints. The splitting of the degrees of freedom into interior and interface parts given by (2.3) induces a reordering

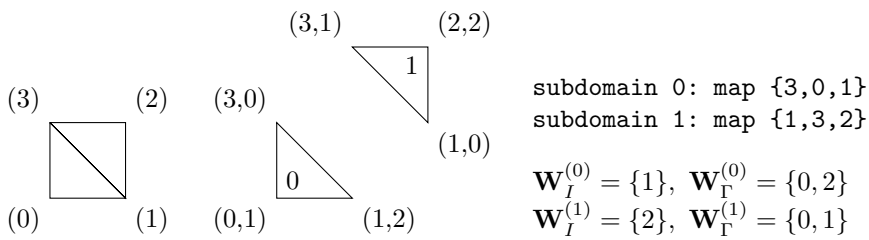

FIG. 1. `MATIS` *mapping example.*

of the global linear operator

$$\widehat{A} = \begin{bmatrix} A_{II} & A_{I\Gamma} \\ A_{\Gamma I} & \widehat{A}_{\Gamma\Gamma} \end{bmatrix} \tag{3.1}$$

that is suitable for a block factorization

$$\widehat{A}^{-1} = \begin{bmatrix} I & -A_{II}^{-1}A_{I\Gamma} \\ 0 & I \end{bmatrix} \begin{bmatrix} A_{II}^{-1} & 0 \\ 0 & \widehat{S}^{-1} \end{bmatrix} \begin{bmatrix} I & 0 \\ -A_{\Gamma I}A_{II}^{-1} & I \end{bmatrix}, \tag{3.2}$$

where

$$\widehat{S} = \widehat{A}_{\Gamma\Gamma} - A_{\Gamma I}A_{II}^{-1}A_{I\Gamma} \tag{3.3}$$

is the interface Schur complement, which is never assembled explicitly, since it can be obtained by subassembling

$$\widehat{S} = R_\Gamma^T S R_\Gamma$$

with $R_\Gamma^{(i)}$ the restriction operator from $\widehat{\mathbf{W}}_\Gamma$ to $\mathbf{W}_\Gamma^{(i)}$, R_Γ the direct sum of the $R_\Gamma^{(i)}$, and S the block diagonal matrix of local Schur complements

$$S = \begin{bmatrix} S^{(1)} & & \\ & \ddots & \\ & & S^{(N)} \end{bmatrix}, \tag{3.4}$$

where

$$S^{(i)} = A_{\Gamma\Gamma}^{(i)} - A_{\Gamma I}^{(i)}A_{II}^{(i)-1}A_{I\Gamma}^{(i)} \tag{3.5}$$

with $A_{II}^{(i)}$, $A_{I\Gamma}^{(i)}$, $A_{\Gamma I}^{(i)}$, and $A_{\Gamma\Gamma}^{(i)}$ obtained by reordering the local basis functions in interior and interface parts.

BDDC methods were first presented as preconditioners for the Schur complement matrix (3.3), and they can be viewed as a two-level additive alternative to the balancing Neumann–Neumann method [60] (also known as the BDD method). However, any Schur complement preconditioning technique can be recast as a preconditioner for the global problem (3.1) exploiting the block factorization (3.2); different variants of the preconditioners can then be designed by replacing the inverses appearing in formula (3.2) with suitable solvers,

$$M^{-1} = \begin{bmatrix} I & -M_{II}^{-1}A_{I\Gamma} \\ 0 & I \end{bmatrix} \begin{bmatrix} M_{II}^{-1} & 0 \\ 0 & M_\Gamma^{-1} \end{bmatrix} \begin{bmatrix} I & 0 \\ -A_{\Gamma I}M_{II}^{-1} & I \end{bmatrix}. \tag{3.6}$$

In their standard formulation, both BDDC and BDD methods use exact subdomain solvers for the interior Dirichlet problems (i.e., $M_{II}^{-1} = A_{II}^{-1}$) and exploit the subassembled structure of the linear operator to construct scalable preconditioners M_Γ that are also robust with respect to jumps in the coefficient of the PDE aligned with Γ. BDD methods precondition the Schur complement by using solvers with local subdomain matrices (Neumann problems) and by multiplicatively combining these Neumann solvers with a coarse solver, which provides scalability of the methods and well-posedness of the local corrections. On the other hand, the BDDC methods consider a partially assembled Schur complement that is cheaper to invert than $\widehat{S}$; a block factorization of the partially assembled matrix then leads to the additive split of the preconditioner in local and coarse parts.

For additional details on the nonoverlapping domain decomposition framework and on BDD methods, see the monographs [64, 74, 79]; for the BDDC algorithm, see [24, 25, 51, 54, 62] and references therein. For additional user customization and command line options that are not described in what follows in the paper, please consult the online PETSc documentation [9].

3.1. Interface equivalence classes. The first step in BDDC methods consists in the analysis of the interface among subdomains, since the partition of degrees of freedom on Γ into equivalence classes plays a central role in the design, analysis, and programming of the algorithm. Faces (F), edges (E), and vertices (V) classes are detected during the setup phase of `PCBDDC` by using the following rules (see also [48]):

$$\text{2D}\begin{cases} x,y \in E_k \iff |\mathcal{N}_x| = 2,\ \mathcal{N}_x = \mathcal{N}_y,\ x \sim y,\ x \notin \partial\Omega_{N,D}, \\ x \in V_k \iff |\mathcal{N}_x| > 2,\ x \notin \partial\Omega_D, \\ \qquad\qquad \text{or} \\ \qquad\qquad |\mathcal{N}_x| = 2,\ x \in \partial\Omega_N, \end{cases}$$

$$\text{3D}\begin{cases} x,y \in F_k \iff |\mathcal{N}_x| = 2,\ \mathcal{N}_x = \mathcal{N}_y,\ x \sim y,\ x \notin \partial\Omega_{N,D}, \\ x,y \in E_k \iff |\mathcal{N}_x| > 2,\ \mathcal{N}_x = \mathcal{N}_y,\ x \sim y,\ x \notin \partial\Omega_{N,D}, \\ \qquad\qquad \text{or} \\ \qquad\qquad |\mathcal{N}_x| = 2,\ \mathcal{N}_x = \mathcal{N}_y,\ x \sim y,\ x \in \partial\Omega_N, \\ x \in V_k \iff \nexists y \neq x \text{ s.t. } \mathcal{N}_x = \mathcal{N}_y,\ x \sim y, x \notin \partial\Omega_D, \end{cases}$$

where x, y are two different degrees of freedom, $\mathcal{N}_x$ is the set of subdomains sharing x, $\sim$ is a connectivity relation, and $\partial\Omega_N$ (resp., $\partial\Omega_D$) is the part of the boundary where natural (essential) boundary conditions have been imposed. In other words, two degrees of freedom belong to the same class if they are connected and shared by the same set of subdomains. The cardinality of the sharing set discriminates between faces and edges in three dimensions. Finally, a degree of freedom not connected to any other is denoted as a vertex.

The default connectivity between degrees of freedom is induced by the sparsity structure of the subdomain matrices; the use of the default connectivity can be turned off by using the switch `-pc_bddc_use_local_mat_graph 0`, or it can be specified by the user through the function `PCBDDCSetLocalAdjacencyGraph`. Additional information on the degrees of freedom can be provided by the user to further customize the definition of $\sim$. The number of fields and the associated degrees of freedom can be specified via `PCBDDCSetDofsSplitting`, or they can be deduced by `PCBDDC` by detecting the block size of the problem; interface classes will be then made up by degrees of freedom belonging to the same field. Finally, since the current implementation does not support any automatic selection of primal vertices (see [72] and references therein), additional user-defined vertices can be specified in local ordering by using `PCBDDCSetPrimalVerticesLocalIS`.

Essential and natural parts of the domain boundary $\partial\Omega_D$ and $\partial\Omega_N$ (which with abuse of terminology have been denoted respectively by Dirichlet and Neumann) can be specified in global numbering by using `PCBDDCSetDirichletBoundaries` and `PCBDDCSetNeumannBoundaries` or in local subdomainwise ordering via `PCBDDCSetDirichletBoundariesLocal` and `PCBDDCSetNeumannBoundariesLocal`. The boundary specification is not mandatory, but it can improve the convergence properties of the algorithm: in particular, the residual on the degrees of freedom

where essential boundary conditions have been imposed is eliminated at the beginning of the Krylov iterations.

3.2. Primal and dual spaces. At the core of the BDDC methods is the definition of a partially assembled space $\widetilde{\mathbf{W}}_\Gamma$,

$$\widehat{\mathbf{W}}_\Gamma \subset \widetilde{\mathbf{W}}_\Gamma \subset \mathbf{W}_\Gamma,\ \widetilde{\mathbf{W}}_\Gamma = \mathbf{W}_\Delta \oplus \widehat{\mathbf{W}}_\Pi,$$

where $\widehat{\mathbf{W}}_\Pi$ is the *primal* space and $\mathbf{W}_\Delta$ is the product space of the local *dual* spaces

$$\mathbf{W}_\Delta = \prod_{i=1}^{N} \mathbf{W}_\Delta^{(i)}$$

consisting of functions with zero values at the primal degrees of freedom. The functions in the space $\widetilde{\mathbf{W}}_\Gamma$ will be continuous at the (coarse) primal level and discontinuous elsewhere across the subdomain interface. The balancing procedure for BDDC methods consists in properly selecting the primal degrees of freedom in order to have well-posed dual corrections. In addition, the proper choice of primal degrees of freedom contributes to the scalability of the method, mainly intended here as a number of preconditioned Krylov iterations independent of the number of subdomains.

The primal space is typically spanned by the minimum energy extensions of selected degrees of freedom such as subdomain vertices and certain functionals over edges or faces of the subdomains. The continuity of edge and face functionals can be imposed either by using a saddle point formulation [24, 25] or by partial assembling if a change of basis has been performed [51]. Since the two techniques provide equivalent methods, and in order to simplify the description of the algorithm, we assume that a change of basis has been performed and all primal degrees of freedom have been made explicit. We can then reorder the local matrices accordingly, i.e.,

$$A^{(i)} = \begin{bmatrix} A_{II}^{(i)} & A_{I\Delta}^{(i)} & A_{I\Pi}^{(i)} \\ A_{\Delta I}^{(i)} & A_{\Delta\Delta}^{(i)} & A_{\Delta\Pi}^{(i)} \\ A_{\Pi I}^{(i)} & A_{\Pi\Delta}^{(i)} & A_{\Pi\Pi}^{(i)} \end{bmatrix}.$$

Without any additional information from the user, `PCBDDC` uses arithmetic averages for edge and face functionals; these functionals can be customized by attaching a `MatNullSpace` object to the preconditioning matrix via `MatSetNearNullSpace`. The vectors of the `MatNullSpace` object should contain the functionals in the form of quadrature weights; weights for different classes can coexist in the same vector. If more than one functional per class is prescribed, any linear dependence is eliminated by means of edge or face based SVD operations. As an example, the rigid body modes represent a robust set of functionals for the linear elasticity problem, even in the almost incompressible limit [67]. The use of vertices, edges, and faces can be enabled or disabled with the command line switches `-pc_bddc_use_vertices`, `-pc_bddc_use_edges`, and `-pc_bddc_use_faces`.

The PETSc class `PCBDDC` imposes the continuity of primal constraints using the saddle point formulation [24, 25]; the change of basis approach [51] is implemented using a QR factorization of the constraints and it is available via the command line switches `-pc_bddc_use_change_of_basis` and `-pc_bddc_use_change_on_faces`. For certain discretization techniques like the Nédélec elements, the BDDC method needs a specific change of variables which depends on the underlying mesh [26]; in such

cases, the user can specify the desired change of variables attaching a matrix to the `PCBDDC` object using `PCBDDCSetChangeOfBasisMat`, where the columns of the matrix correspond to the representation of the new basis functions in the original basis.

3.3. Schur complement preconditioner. The interface preconditioner for (3.6) is given in terms of the inverse of the partially assembled Schur complement on $\widetilde{\mathbf{W}}_\Gamma$ combined with a proper scaling of interface degrees of freedom, i.e.,

$$M^{-1}_{\text{BDDC}} = \widetilde{R}^T_{D,\Gamma}\widetilde{S}^{-1}\widetilde{R}_{D,\Gamma}, \quad \widetilde{S} = \widetilde{R}_\Gamma S \widetilde{R}^T_\Gamma, \tag{3.7}$$

where S is given by (3.4). Additional restriction operators need to be defined to complete the description of the BDDC algorithm,

$$\begin{aligned} R^{(i)}_\Delta &: \mathbf{W}_\Delta \to \mathbf{W}^{(i)}_\Delta, \quad R^{(i)}_\Pi : \widehat{\mathbf{W}}_\Pi \to \mathbf{W}^{(i)}_\Pi, \\ R_{\Gamma\Delta} &: \mathbf{W}_\Gamma \to \mathbf{W}_\Delta, \quad R_{\Gamma\Pi} : \mathbf{W}_\Gamma \to \widehat{\mathbf{W}}_\Pi, \end{aligned}$$

with $\widetilde{R}_\Gamma = R_{\Gamma\Pi} \oplus R_{\Gamma\Delta}$.

The scaling operator $\widetilde{R}_{D,\Gamma}$ plays a central role in the algorithm and it is described in more detail in section 3.4. The matrix $\widetilde{S}$ is usually not assembled explicitly; instead, the inverse of $\widetilde{S}$ is obtained by block elimination [54],

$$\widetilde{S}^{-1}_\Gamma = R^T_{\Gamma\Delta}\left(\sum_{i=1}^N \begin{bmatrix} 0 & R^{(i)T}_\Delta \end{bmatrix} \begin{bmatrix} A^{(i)}_{II} & A^{(i)}_{I\Delta} \\ A^{(i)}_{\Delta I} & A^{(i)}_{\Delta\Delta} \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ R^{(i)}_\Delta \end{bmatrix}\right) R_{\Gamma\Delta} + \Phi S^{-1}_{\Pi\Pi}\Psi^T, \tag{3.8}$$

which expresses the application of the BDDC preconditioner as the additive combination of local subdomain corrections and a global coarse correction; the latter imposes the continuity on the primal space and provides the global exchange of information between subdomains that is required to obtain scalable preconditioners in terms of the number of Krylov iterations. The matrices defining the primal basis functions are the minimal energy extension of the primal degrees of freedom into the subdomains

$$\begin{aligned} \Phi =& R^T_{\Gamma\Pi} - R^T_{\Gamma\Delta}\sum_{i=1}^N \left(\begin{bmatrix} 0 & R^{(i)T}_\Delta \end{bmatrix} \begin{bmatrix} A^{(i)}_{II} & A^{(i)}_{I\Delta} \\ A^{(i)}_{\Delta I} & A^{(i)}_{\Delta\Delta} \end{bmatrix}^{-1} \begin{bmatrix} A^{(i)}_{I\Pi} \\ A^{(i)}_{\Delta\Pi} \end{bmatrix} R^{(i)}_\Pi\right), \\ \Psi =& R^T_{\Gamma\Pi} - R^T_{\Gamma\Delta}\sum_{i=1}^N \left(\begin{bmatrix} 0 & R^{(i)T}_\Delta \end{bmatrix} \begin{bmatrix} A^{(i)}_{II} & A^{(i)}_{I\Delta} \\ A^{(i)}_{\Delta I} & A^{(i)}_{\Delta\Delta} \end{bmatrix}^{-T} \begin{bmatrix} A^{(i)T}_{\Pi I} \\ A^{(i)T}_{\Pi\Delta} \end{bmatrix} R^{(i)}_\Pi\right). \end{aligned} \tag{3.9}$$

In the case of symmetric problems, the computation of Ψ is not needed. The primal coarse problem is defined as the projection of $\widetilde{S}$ on the primal basis functions and it can be assembled by considering individual subdomain contributions,

$$S_{\Pi\Pi} = \Psi^T\widetilde{S}\Phi = \sum_{j=1}^N R^{(i)T}_\Pi S^{(i)}_{\Pi\Pi} R^{(i)}_\Pi, \tag{3.10}$$

with each contribution computed in terms of local Schur complements with respect to the primal degrees of freedom, i.e.,

$$S^{(i)}_{\Pi\Pi} = A^{(i)}_{\Pi\Pi} - \begin{bmatrix} A^{(i)}_{\Pi I} & A^{(i)}_{\Pi\Delta} \end{bmatrix} \begin{bmatrix} A^{(i)}_{II} & A^{(i)}_{I\Delta} \\ A^{(i)}_{\Delta I} & A^{(i)}_{\Delta\Delta} \end{bmatrix}^{-1} \begin{bmatrix} A^{(i)}_{I\Pi} \\ A^{(i)}_{\Delta\Pi} \end{bmatrix}. \tag{3.11}$$

No additional computations are needed to compute (3.11), since it can be obtained as a by-product of the computation of the primal basis functions (3.9). It must be noted that the application of the interface preconditioner (3.8) and the computation of the primal basis functions (3.9) are slightly different when a saddle point formulation is used for imposing the primal constraints. To save space, we omit here their description; additional details can be found in [25].

Turning to the PETSc implementation, command line switches are available to experiment with different solvers for the static condensation (Dirichlet) step M_{II}^{-1} in (3.6) and for the subdomain Neumann correction in (3.8) using the option prefixes `-pc_bddc_dirichlet_` and `-pc_bddc_neumann_`. Solvers for the coarse problem can be selected at run-time by using the option prefix `-pc_bddc_coarse_`. For nonsymmetric problems, the computation of Ψ in (3.9) must be enabled by using the command line switch `-pc_bddc_symmetric 0`.

An extension of the BDDC algorithm to use approximate solvers has been proposed in [25], together with an interesting algebraic variant of (3.6), available in the `PCBDDC` class by using the command line switch `-pc_bddc_switch_static`. The function `PCBDDCSetNullSpace` should be used to specify the kernel of the local subdomain matrices needed to compute the local nullspace corrections of inexact subdomain solvers in BDDC methods [25].

The setup of parallel solvers for the coarse problem can be a bottleneck when a large number of subdomains and/or many primal constraints per subdomain are considered. Also, the parallel backward and forward substitutions can affect the timings of the application of the preconditioner with a large number of subdomains. In fact, even if the coarse problem matrix is sparse, the number of nonzeros per row can be quite large, since it is obtained by assembling dense subdomain contributions as given by (3.11); in turn, the memory requirements for the factorization step of parallel direct solvers could become prohibitive. Different strategies have been implemented in the `PCBDDC` class in order to overcome the latter issues:

- If some of the MPI processes of the `MATIS` object's communicator are not associated with any subdomain (i.e., the local size of their local to global map in the `MATIS` constructor is zero), then `PCBDDC` automatically maps the coarse problem on those MPI processes. This approach has the advantage that the subdomain corrections and the solution of the primal problem can be overlapped during the application of the preconditioner [7].
- In order to deal with memory issues of parallel direct solvers, the command line switch `-pc_bddc_coarse_redistribute` is available to redistribute the coarse problem on a subcommunicator; an additional integer parameter to the previous switch defines the size of the subcommunicator. As a result of the remap, the coarse matrix will have a larger number of local (per MPI process) rows and thus the amount of communications needed by a parallel direct coarse solver will be smaller.
- Given the unassembled nature of the coarse problem (3.10) and the algebraic nature of the BDDC algorithm, a multilevel extension is readily available [80, 81]. In the multilevel BDDC framework, the solution of the coarse problem is replaced by the application of a BDDC preconditioner at a coarser level. The use of approximate coarse solvers could increase the total number of iterations of the Krylov solver [25] but also lead to highly scalable preconditioners. Additional details on the current implementation of multilevel BDDC are given in section 3.6. Large-scale numerical results are provided in section 4.

3.4. Scaling operator. The scaling operator appearing in (3.7) is defined as

$$\widetilde{R}_{D,\Gamma} = R_{\Gamma\Pi} \oplus R_{D,\Delta} R_{\Gamma\Delta},\ R_{D,\Delta} = \oplus_{i=1}^{N} D^{(i)} R_{\Delta}^{(i)},$$

where $D^{(i)}$ are subdomain matrices defined to realize the following partition of unity:

$$E_D \mathbf{w} = \widetilde{R}_{\Gamma} \widetilde{R}_{D,\Gamma}^{T} \mathbf{w} = \mathbf{w} \ \forall\ \mathbf{w} \in \widehat{\mathbf{W}}_{\Gamma}, \tag{3.12}$$

where E_D is the *average operator* induced by the scaling. The proper choice of scaling guarantees the robustness of the method with respect to jumps in the coefficients of the PDE aligned with the interface between the subdomains [61]; for a proof of the robustness of BDDC with more general coefficient distributions see [70, 71].

For scalar elliptic PDEs with only one material coefficient, the scaling matrices $D^{(i)}$ are defined as diagonal matrices, with the diagonal entry corresponding to the degree of freedom x given as a weighted average of subdomain values $\rho_x^{(i)}$ of x as

$$d_x^{(i)} = \frac{\rho_x^{(i)}}{\sum_{j \in \mathcal{N}_x} \rho_x^{(j)}} \tag{3.13}$$

with $\mathcal{N}_x$ the set of subdomains sharing x (see [79]); this scaling is also valid for linear elasticity in the compressible case, where a robust choice in case of subdomainwise constant coefficients corresponds to using one of the Lamé parameters, i.e., $\rho_x^{(i)} = \mu^{(i)}$ [51]. The so-called stiffness scaling is obtained by setting $\rho_x^{(i)}$ equal to the diagonal entry of the subdomain matrix relative to x. An optimal choice for $\rho_x^{(i)}$ depends on the problem and on the distribution of the coefficients and it is thus left to the user. For some possible solutions, see [71].

An advanced scaling operator, named *deluxe*, has been recently proposed to deal with the case of jumping coefficients for systems of PDEs with more than one material parameter [86]. Such scaling has also proven very effective for unstructured grid computation with irregular subdomains given by mesh partitioners [18]. In the deluxe case, the scaling matrices $D^{(i)}$ are block diagonal, with dense blocks corresponding to each class of the local interface. Given a face or an edge F of a subdomain, the diagonal blocks accounting for the scaling of the degrees of freedom on F are given by

$$\left(\sum_{j \in \mathcal{N}_F} S_F^{(j)} \right)^{-1} S_F^{(i)}, \tag{3.14}$$

where $S_F^{(i)}$ is the principal minor of $S^{(i)}$ given in (3.5) relative to F and $\mathcal{N}_F$ the set of subdomains sharing F.

Since the entries of $S^{(i)}$ are not readily available, different strategies have been implemented in `PCBDDC` to compute the action of the deluxe operator on a given vector. The basic strategy consists in repeatedly applying $S^{(i)}$ to unit vectors in order to compute each subdomain block $S_F^{(i)}$. This strategy leads to noncompetitive algorithms, since a Dirichlet problem has to be solved for each degree of freedom of F; however, the computational costs can be greatly reduced using the *economical* version of the scaling [26], which considers Schur complements with respect to a thin neighborhood of F rather than the whole union of subdomains having F in common.

A different implementation of deluxe scaling exploits one of the most recent features of sequential factorization packages, which consists in the ability of providing the factors of $A_{II}^{(i)-1}$ and the explicit form of $S^{(i)}$ when factoring $A^{(i)}$ at some additional cost (see, e.g., [3]). Irrespective of the technique used, once matrices $S_F^{(i)}$ are available, only nearest neighbor communications are needed to assemble the sum of Schur complements appearing in formula (3.14); the factorization of the latter matrix can then be carried out separately on each subdomain.

The `PCBDDC` class uses by default the *cardinality scaling*, which is obtained by using $\rho_x^{(i)} = 1$ in (3.13). A subdomainwise constant value for $\rho_x^{(i)}$ can be specified by using `PCISSetSubdomainScalingFactor`. More general pointwise scaling can be defined by using `PCISSetSubdomainDiagonalScaling`; the stiffness scaling is available by using the command line switch `-pc_is_use_stiffness_scaling`; deluxe scaling is available by using the command line switch `-pc_bddc_use_deluxe_scaling`. The command line switch `-pc_bddc_schur_layers` switch on the economic version of the scaling by defining the number of layers of degrees of freedom which have to be included in the explicit computation of the local Schur complements. The current `PCBDDC` implementation uses the external package MUMPS [3] for the explicit computation of the local Schur complements; since PETSc has to be linked against external libraries supporting 64-bit integer representation in order to solve problems with more than 2.1 billion distributed unknowns, the current deluxe version of `PCBDDC` with explicit Schur complement support is limited to 32-bit integer builds, since the current version of MUMPS does not support parallel builds with 64-bit integers.

3.5. Adaptive selection of constraints. Although the proper choice of the scaling operator ensures robustness of the BDDC methods with respect to jumps in the PDE coefficients aligned with the interface, the convergence rate of the associated preconditioned Krylov methods usually deteriorates when such jumps are not aligned with the interface. After the pioneering work [63], in recent years different approaches have been proposed to accommodate arbitrary jumps in the coefficients of elliptic PDEs within BDDC methods [15, 19, 41, 42, 44, 45, 46, 69, 75].

The approach proposed in [75] selects face constraints by iteratively solving sparse eigenproblems defined on each pair of subdomains sharing a face and its boundary. Edge primal constraints, which could be obtained as a by-product of the eigensolver, were not considered in the numerical experiments due to a loss of sparsity of the projected operators involved. An extension to edge constraints has been recently analyzed in the case of compressible and almost incompressible linear elasticity [44].

On the other hand, uncoupled dense generalized eigenproblems defined on each face or edge are instead considered in [15, 19, 41, 42, 45, 46, 69]; some differences exist among these techniques, but their common feature set consists in working with a collection of relatively small, sequential, and dense data structures instead of using iterative eigensolvers and PDE-dependent projection operators. This makes them appear attractive for future exascale machines or for targeting hardware accelerators.

The PETSc class `PCBDDC` implements the adaptive approach first proposed in [69], by combining constraint selection with the optimal deluxe scaling, which in turn permits us to reduce BDDC condition number estimates to individual bounds for subdomains [86]; also, there is evidence that the adaptive primal spaces generated by deluxe BDDC algorithms are smaller than those generated by using pointwise scalings [42]. The chosen approach has the further advantage that only one generalized eigenvalue problem has to be solved for each interface class, whereas the approaches considered in [41, 45] need the solution of two different eigenproblems.

When dealing with SPD problems, the norm of the average operator given by (3.12) is used to estimate the condition number of the BDDC preconditioned system [62]; the estimation is then usually performed by analyzing vertex, edge, and face contributions separately. Using deluxe scaling, face estimation in three dimensions or edge estimation in two dimensions can be reduced to the analysis of the generalized eigenvalue problem

$$(S_F^{(i)} : S_F^{(j)})\boldsymbol{\phi} = \lambda(\widetilde{S}_F^{(i)} : \widetilde{S}_F^{(j)})\boldsymbol{\phi} \tag{3.15}$$

with F a given class of the interface shared by subdomains i and j, the : operator defined for SPD matrices as

$$S_F^{(i)} : S_F^{(j)} = (S_F^{(i)-1} + S_F^{(j)-1})^{-1}$$

and

$$\widetilde{S}_F^{(i)} = S_{FF}^{(i)} - S_{F'F}^{(i)T} S_{F'F'}^{(i)-1} S_{F'F}^{(i)}, \tag{3.16}$$

with $F' = \Gamma^{(i)} \setminus F$; the matrices $S_{FF}^{(i)}$, $S_{F'F}^{(i)}$, and $S_{F'F'}^{(i)}$ are obtained by reordering the local Schur complement matrix accordingly.

If we include in the primal space all the vectors of the form $(S_F^{(i)} : S_F^{(j)})\boldsymbol{\phi}_k$, where $\boldsymbol{\phi}_k$ are generalized eigenvectors of (3.15) corresponding to the eigenvalues greater than a given threshold λ_m, then the contribution of F to the maximum eigenvalue of the preconditioned operator will be less than λ_m times a constant [19, 42, 46].

A careful inspection of (3.16) reveals that $\widetilde{S}_F^{(i)}$ will not be in general positive definite, being the Schur complement of $S^{(i)}$, which could be positive semidefinite depending on the PDE, the geometry of the subdomains, and the boundary conditions. The strategy currently implemented in `PCBDDC` overcomes the possible positive semidefinitness of the local Schur complements by constructing the $\widetilde{S}_F^{(i)}$ matrices from

$$S_r^{(i)} = A_{rr}^{(i)} - A_{Ir}^{(i)T} A_{II}^{(i)-1} A_{Ir}^{(i)}, \quad r = \Gamma_i \setminus V,$$

instead of $S^{(i)}$, where V is the set of primal vertices for subdomain Ω_i, since each $S_r^{(i)}$ is well-defined by construction.

The computational costs of assembling and solving the generalized eigenvalue problem (3.15) can be greatly reduced, since we can directly use the generalized eigenvectors corresponding to the largest eigenvalues of

$$\left(\widetilde{S}_F^{(i)-1} + \widetilde{S}_F^{(j)-1}\right)\boldsymbol{\phi} = \lambda\left(S_F^{(i)-1} + S_F^{(j)-1}\right)\boldsymbol{\phi}, \tag{3.17}$$

resulting in the saving of two explicit matrix inversions and one dense matrix-matrix multiplication for each class of the local interface.

The adaptive selection of constraints for BDDC using deluxe-based estimates is still an active topic of research for edge classes; for very recent results, see [19, 42, 46]: differently from these approaches, `PCBDDC` implements

$$\left(\sum_{j\in\mathcal{N}_E} \widetilde{S}_E^{(j)-1}\right)\boldsymbol{\phi} = \lambda\left(\sum_{j\in\mathcal{N}_E} S_E^{(j)-1}\right)\boldsymbol{\phi}, \tag{3.18}$$

which is a heuristic approach that generalizes (3.17) to edge classes. Adaptivity for symmetric indefinite problems or more general operators could be the subject of future research.

The current implementation uses dense linear algebra kernels provided by LAPACK [4] to solve (3.17) and (3.18) for each class of the local interface. The small dense blocks $S_F^{(i)}$ are inverted explicitly. Formula (3.16) is not used in practice; instead, each local Schur complement is inverted explicitly, and then the principal minors $\widetilde{S}_F^{(i)-1}$ are extracted. Such an approach has the further advantage that the inverted matrices can be reused when solving the local corrections in (3.8) and in (3.9) by means of dense matrix-vector products. Since MUMPS is used to explicitly compute the Schur complements, the current implementation is limited to 32-bit integer builds.

3.6. Multilevel extensions. The definition of the BDDC coarse problem given in (3.10) naturally leads to a multilevel extension of the algorithm where a subdomain at the fine level is considered an element of a coarser mesh, with coarse element matrices given by (3.11) [80, 81]; the solution of the coarse problem at a given level is then replaced by the application of a BDDC preconditioner defined on the coarser level. In the current implementation, multilevel BDDC can be selected by using the command line switch `-pc_bddc_levels n`, where `n` is the number of additional levels requested. Options for the solvers at coarser levels can be specified at the command line by using the prefix `-pc_bddc_lm_`, where `m` should be replaced by the desired level number. The command line switch `-pc_bddc_coarsening_ratio` controls the number of coarse elements which are aggregated into a coarse subdomain at the coarser levels; any of the graph partitioning packages interfaced to PETSc can be used to compute the aggregation. The command line switch `-pc_bddc_coarse_adj` can be used to specify the number of processes used within the partitioning procedure, in order to increase the quality of the partitioning and to reduce its computational costs.

The current `PCBDDC` implementation exploits the additive nature of the application of the preconditioner (3.8) and employs subcommunicators for the coarser levels of the hierarchy, in order to overlap subdomain and coarse solvers at a given level. Preconditioner setup is instead performed sequentially with respect to the number of levels. Figure 2 contains a schematic diagram of how the MPI processes are mapped throughout the multilevel BDDC hierarchy. The example considers eight subdomains at the fine level, one for each MPI process; two additional levels are considered with a coarsening ratio of 2. MPI process placement for the finest and coarser subdomains is schematically drawn on the left and on the right, respectively. On each level, subdomains are labeled by the couple (rank, level), where rank is the MPI rank in the communicator associated with the level. Arrows indicate the aggregation scheme in the coarsening procedure. With regard to the example considered, substructure corrections are first applied at the finest level (0); local corrections at levels 1 and 2 are then applied concurrently to the solution of the coarse problem on level 2, which is mapped on the MPI processes labeled by $S_{\Pi\Pi}$ in the figure.

3.7. FETI-DP methods. The `PCBDDC` class provides experimental support for FETI-DP methods, exploiting the well-known duality between BDDC and FETI-DP algorithms [62]. Briefly, FETI-DP methods solve the following minimization problem [29]:

$$\operatorname*{argmin}_{\mathbf{w}\in\widetilde{\mathbf{W}}} \left[\frac{1}{2}\mathbf{w}^T\widetilde{A}\mathbf{w} - \mathbf{w}^T\widetilde{\mathbf{f}}\right] \text{ s.t. } B\mathbf{w} = 0, \tag{3.19}$$

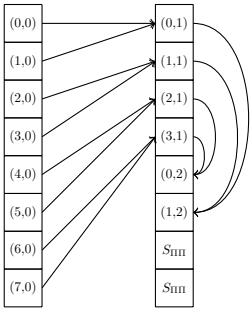

FIG. 2. *Schematic diagram of process distribution and preconditioner application for multilevel BDDC.*

with $\widetilde{A}$ the linear operator (2.1) partially assembled at primal degrees of freedom and $\widetilde{\mathbf{f}}$ the partially assembled right-hand side; the jump operator

$$B = \left[B^{(1)}|\dots|B^{(N)}\right], \quad B\mathbf{w} = 0 \iff \mathbf{w} \in \widehat{\mathbf{W}},$$

imposes the continuity of dual degrees of freedom between subdomains. After the introduction of the saddle point formulation associated to (3.19),

$$\begin{bmatrix} \widetilde{A} & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} \mathbf{w} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \widetilde{\mathbf{f}} \\ \mathbf{0} \end{bmatrix}, \quad \boldsymbol{\lambda} \in \Lambda = \text{Range}(B),$$

the FETI-DP linear system for the set of Lagrange multipliers is obtained by statically condensing the degrees of freedom on $\widetilde{\mathbf{W}}$ as

$$F\boldsymbol{\lambda} = \mathbf{d}, \quad F = B\widetilde{A}^{-1}B^T, \quad \mathbf{d} = B\widetilde{A}^{-1}\widetilde{\mathbf{f}}. \tag{3.20}$$

Note that primal continuity is enforced by partial assembling in FETI-DP, whereas the continuity for dual degrees of freedom is obtained at the convergence of the method; alternative approaches based on projector preconditioning have been proposed for imposing primal continuity within FETI-DP methods [47].

The application of $\widetilde{A}^{-1}$ involves a block inversion technique and it can be obtained by reusing the computational routines for the application of the interface BDDC preconditioner (3.8). Once (3.20) has been solved, the solution in the original discrete space can be obtained as

$$\mathbf{w} = \widetilde{A}^{-1}\left(\widetilde{\mathbf{f}} - B^T\boldsymbol{\lambda}\right), \quad \mathbf{w} \in \widehat{\mathbf{W}}.$$

An almost optimal preconditioner for (3.20) is the *Dirichlet preconditioner*

$$M_D^{-1} = B_D\overline{R}_\Delta^T \left(\sum_{i=1}^{N} R_\Delta^{(i)}(A_{\Delta\Delta}^{(i)} - A_{I\Delta}^{(i)T}A_{II}^{(i)-1}A_{I\Delta}^{(i)})R_\Delta^{(i)}\right)\overline{R}_\Delta B_D^T,$$

with $\overline{R}_\Delta$ the restriction operator from $\widetilde{\mathbf{W}}_\Gamma \oplus \mathbf{W}_I$ to $\mathbf{W}_\Delta$ and B_D the scaled jump operator, which is built combining B with the diagonal scaling matrices $D^{(i)}$ defined in section 3.4; for additional details see [79].

In the current implementation, the FETI-DP matrix F and the Dirichlet preconditioner M_D^{-1} can be obtained via `PCBDDCCreateFETIDPOperators`. A suitable

right-hand side for FETI-DP can be computed from the original right-hand side via `PCBDDCMatFETIDPGetRHS`; the solution in the original discrete space can be obtained from the FETI-DP solution by calling `PCBDDCMatFETIDPGetSolution`. Nonredundant Lagrange multipliers are implemented by default when assembling the jump operator; fully redundant Lagrange multipliers can be requested via the command line switch `-fetidp_fullyredundant`. The scaled jump operators are then constructed in accordance with the selected type of multipliers [88]. Deluxe scaling and constraints adaptivity are not yet supported; a specific FETI-DP class will be designed in the future PETSc releases with additional support for inexact methods [49].

4. Numerical results. In this section, we present numerical results related to the multilevel extension of `PCBDDC` (see section 3.6) and to the adaptive selection of constraints (see section 3.5) implemented in the current version of PETSc [10]; large-scale numerical results are also provided combining adaptivity and multileveling. Numerical results using the previous versions of the code appeared elsewhere: BDDC and FETI-DP methods applied to symmetric positive semidefinite linear systems arising in cardiac electrophysiology have appeared in [88]; an extension to approximate subdomain solvers has been studied in [87]. Previous versions of the `PCBDDC` code also have been applied to the spectral element discretization of almost incompressible elasticity in three dimensions [67] and within the isogeometric analysis framework [13, 14]. Finally, a Newton–Krylov–BDDC approach has been recently proposed for nonlinear mechanical models of cardiac tissue deformation [68].

The numerical results provided in this section have been obtained on the new Cray XC40 Shaheen of KAUST [34], which features 6192 dual 16-core Haswell processors clocked at 2.3 GHz and equipped with 128 GB of DRAM per node, for a total of 198,144 cores (ranked ninth as of November 2015 in the TOP500 list, http://www.top500.org/).

The domain considered is always $\Omega = [0,1]^3$, SPD linear systems are solved by means of the preconditioned conjugate gradient (PCG) method using a single reduction per iteration and `PCBDDC` as a preconditioner; the right-hand sides are always randomly chosen and the relative residual reduction of 10^{-8} is used as a stopping criterion. PETSc has been compiled with the GNU compiler version 4.9.2, using optimization level `-O3` and with support for AVX instructions. Intel MKL version 11.2.2 has been used for linear algebra kernels.

In what follows, results will be provided for the following three-dimensional model problem: find $\mathbf{u} \in H_0(\mathrm{div}, \Omega)$ such that

$$\int_\Omega (\alpha \ \mathrm{div}\ \mathbf{u}\ \mathrm{div}\ \mathbf{v} + \beta\ \mathbf{u}\cdot\mathbf{v})\ dx = \int_\Omega \mathbf{f}\cdot\mathbf{v}\ dx, \quad \mathbf{v} \in H_0(\mathrm{div}, \Omega), \tag{4.1}$$

where α is a nonnegative function in $L^\infty(\Omega)$, β is a strictly positive function in $L^\infty(\Omega)$, and $\mathbf{f} \in (L^2(\Omega))^3$. The space $H(\mathrm{div}, \Omega)$ consists of vector-valued functions $\mathbf{u}$ such that $\mathbf{u} \in (L^2(\Omega))^3$ and $\mathrm{div}\ \mathbf{u} \in L^2(\Omega)$, with $H_0(\mathrm{div}, \Omega)$ the proper subspace of $H(\mathrm{div}, \Omega)$ with vanishing normal component on $\partial\Omega$. Such a problem arises in first order least-squares formulations of second order elliptic problems [17], regularization or pseudostress-vorticity formulation of the Navier–Stokes equations [56], preconditioning of mixed finite elements [5], and block preconditioning of the Brinkman equations of porous media flow [85]. The same model problem (4.1) has been used to validate a robust auxiliary space multigrid preconditioner for $H(\mathrm{div})$ problems in [52].

The bilinear form given in (4.1) is discretized using the lowest order Raviart–Thomas tetrahedral elements that are conforming in $H(\mathrm{div}, \Omega)$ [16]; the degrees of

freedom are the average values of the normal components of $\mathbf{u}$ over the faces of the elements. A polylogarithmic bound for the condition number using BDDC with deluxe scaling has been proven in [65] under the assumption of piecewise constant distribution of the material coefficients and that each subdomain is the union of a few well-shaped coarse elements of a coarse triangulation; a recipe for the characterization of the primal space of the BDDC operator is also provided. Extensive numerical results for (4.1), complementary to those provided in the next sections, and with challenging three-dimensional coefficient distributions, can be found in [65].

For the numerical results, the triangulation of Ω and the computation of the subassembled matrix are carried out using the C++ library DOLFIN [58], which is part of the FENICS project [57]; ParMETIS is used to decompose the meshes. Deluxe scaling is used throughout all the simulations. MUMPS [3] Cholesky factorizations are used for the subdomain solvers.

4.1. Adaptive coarse spaces for $H(\mathbf{div})$. The first test case is meant to validate the adaptive framework implemented in `PCBDDC`. The domain is discretized by a $48 \times 48 \times 48$ hexahedral grid, with each hexahedron thereafter subdivided into 6 tetrahedra. The tetrahedral mesh is then decomposed into 40 subdomains; the number of degrees of freedom for the test case is 1.3 million. The material coefficients α and β of (4.1) are randomly chosen, element by element, in the ranges $[10^{-p},\ 10^{p}]$ and $[10^{-q},\ 10^{q}]$, respectively; we first draw uniformly distributed random numbers x in the interval $[-p,\ p]$ (resp., $[-q,\ q]$), and then set α (resp., β) to 10^{x}; the threshold considered for the eigenproblem (3.17) is 10. Table 1 shows the iteration counts (it) and condition number estimates (κ_2) for different orders of magnitude in the contrast of the coefficients; contrast in α is increased from left to right, contrast in β from top to bottom. Iteration counts and condition numbers are independent of the contrast of the material coefficients. The condition numbers reported are always smaller than or very close to the prescribed threshold.

The second set of results provides additional insights on the adaptive enrichment of the primal space. The numerical setting is the same as before, except that here the contrast in the material coefficients is kept fixed ($p = q = 3$), and different threshold values for (3.17) are considered. Figure 3 shows the iteration counts (left panel) and the number of iterations (central panel) as a function of the eigenvalue tolerance. As the tolerance decreases, the number of iterations needed by the PCG always decreases; the effectiveness of using smaller tolerances increases as the tolerance decreases, since more constraints are added at lower tolerances. Indeed, approximately 1000 primal constraints are enough to decrease the iterations from 70 to 15; instead,

TABLE 1

Iteration counts (it) and condition number estimates (κ_2) of the PCG method are reported for different values of the contrast in material coefficients. Material coefficients α and β of (4.1) *are randomly chosen in the range $[10^{-p},\ 10^{p}]$ and $[10^{-q},\ 10^{q}]$, respectively. Adaptive BDDC deluxe with eigenvalue threshold* 10.

q \ p	0		1		2		3		4	
	it	κ_2	it	κ_2	it	κ_2	it	κ_2	it	κ_2
0	17	6.05	16	5.85	14	5.13	11	4.03	10	3.58
1	22	9.98	22	9.99	20	9.70	18	10.22	15	8.64
2	22	9.75	22	9.73	20	9.64	19	11.45	18	12.53
3	21	8.58	21	8.55	20	9.26	17	9.45	17	9.05
4	22	9.54	21	9.62	20	9.90	19	10.28	17	9.09

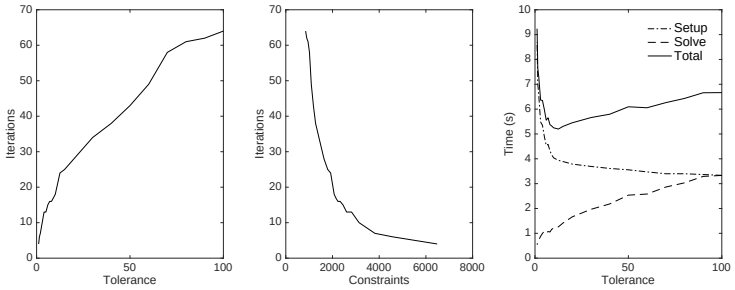

FIG. 3. *PCG iteration counts as a function of eigenvalue tolerance (left panel) and primal constraints (central panel). Right panel shows setup times, solving times for Krylov solver and total (setup+solve) time as a function of eigenvalue tolerance. Fixed distribution of material coefficients of* (4.1) *randomly chosen in* $[10^{-3},\ 10^{3}]$.

almost 4000 constraints are needed to further reduce the number of iterations from 15 to 3. Computational times shown in the right panel are given for the setup of the preconditioner (Setup), for the Krylov solver (Solve), and for the sum of the two phases (Total) as a function of the eigenvalue tolerance; as expected, as the tolerance is decreased, setup times increase and solving times decrease. At larger tolerances (thus with smaller coarse spaces and less effective BDDC preconditioners), the costs associated with the eigenvalue computations and with the factorization of the coarse problem are negligible compared to the setup of the local solvers and to the explicit computations of the Schur complements (data not shown); as a result, setup times are almost constant for thresholds larger than 10. For smaller thresholds, a very rich coarse space is instead generated and factored, and computational times grow accordingly. On the other hand, solving times constantly decrease as the tolerance is decreased, as a result of a smaller number of Krylov iterations; also, the costs associated to the application of the preconditioner remain almost constants (data not shown) for the case considered.

4.2. Weak scaling of adaptive multilevel BDDC. We next report on two weak scaling tests for problem (4.1). The purpose of the first test is to validate the multilevel BDDC framework with high-quality coarse BDDC solvers. Subdomain sizes are kept fixed with approximately 35K degrees of freedom per core while the number of processors is increased from 4096 to 32,768; the number of degrees of freedom ranges from 108 million to 915 million, and the size of the coarse problem increases from 28K to 230K degrees of freedom. Constant material parameters are considered, and the averages of the normal components on each subdomain face (as given by the so-called no-net-flux condition) are used as primal constraints at the finest level [65]. In Figure 4, standard BDDC with MUMPS parallel Cholesky solver on a subcommunicator (direct MUMPS label) is compared against adaptive multilevel BDDC with coarsening ratio 32 (CR32 label); setup (left) and solving (center) times as a function of the number of subdomains are provided, together with the total number of iterations (right) needed by the PCG. For the standard BDDC case, the computational times reported correspond to the best results obtained by testing different sizes of the subcommunicator associated with the coarse problem. In the multilevel case, ParMETIS is used to distribute the coarse mesh, and constraint adaptivity is used at the coarser level, since an optimal set of primal constraints for this problem is not known a priori; an eigenvalue threshold equal to 2 is used. A sequential Cholesky solver from MUMPS

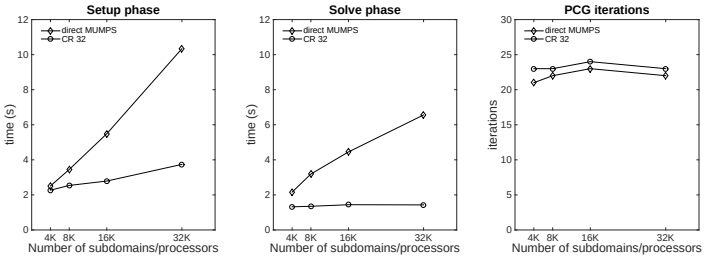

FIG. 4. *Weak scaling test for* (4.1) *with constant material coefficients. Standard BDDC with direct coarse solver (direct MUMPS) is compared against adaptive multilevel BDDC with coarsening ratio* 32 *(CR*32*). Computational times in seconds for the setup of the preconditioner (left) and for the PCG (right) are shown as a function of the number of processors.*

is always used for the coarse problems generated at the coarser levels, given that the sizes of these problems are always small.

Setup times depend on the number of subdomains in both cases; however, the setup costs of the multilevel BDDC preconditioner at the second level are always modest and the dependence on the number of subdomain is weaker than if using a parallel direct coarse solver. In particular, the increase of setup times for the multilevel case is due to the partitioning of the coarser mesh, the redistribution of the coarse elements matrices (3.11), and the following assembling of coarse subdomain matrices; the costs related to the adaptive enrichment of primal space at the coarse level are negligible (data not shown). The solving times using a direct coarse solver degrade as the number of processors increases, since the coarse solver dominates the timings of the preconditioning step. On the other hand, the inexact solution of the coarse problem using a high-quality adaptive BDDC preconditioner does not significantly affect the convergence properties of the method, since it requires only a few more iterations to converge than the exact version. Solving times are almost constant, and the efficiency of the application stage of the adaptive multilevel BDDC preconditioner is 99% with 32K cores, computed as

$$E_p = p_1 \frac{S_p}{p}, \; S_p = \frac{N_{p_1}}{T_{p_1}} \frac{T_p}{N_p}, \tag{4.2}$$

where N_p is the size of the problem solved with p processes, T_p the time spent in the stage, and T_{p_1} the reference time with $p_1 = 4096$ cores.

The second weak scaling test is performed using the same settings as before, except that the material parameters are here elementwise randomly chosen with $p = q = 2$, and the number of the degrees of freedom per subdomain is larger, with each subdomain having approximately 80K unknowns; the total number of degrees of freedom ranges from 263 million to 2.1 billion. Threshold values of 10 and 2 have been used for the generalized eigenvalue problem (3.17) at the finest and coarse levels, respectively. Different coarsening ratios for the adaptive multilevel BDDC (64, 96, and 128) are here compared, whereas the standard BDDC with parallel Cholesky coarse solver has not been considered, since the sizes of the coarse problems generated by the adaptive selection of constraints is much larger than in the previous test; at the finest levels, coarse problem sizes range from 218K for the case with 4K subdomains to 1.9M degrees of freedom for the case with 32K subdomains.

Computational times for the setup and solve phases of `PCBDDC` are reported as a function of the number of subdomains in Figure 5. Setup times for the adaptive

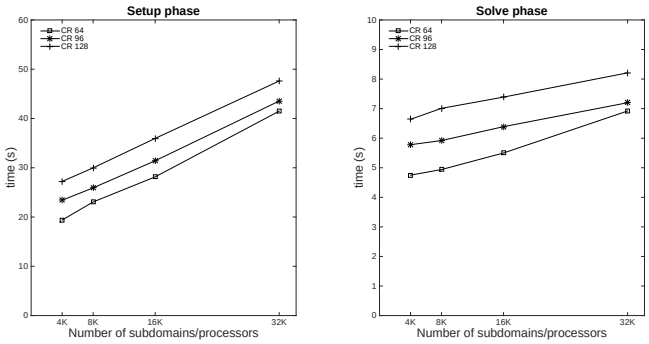

FIG. 5. *Weak scaling test for* (4.1) *with random material coefficients in* $[10^{-2}, 10^2]$. *Adaptive multilevel BDDC with coarsening ratios* 64 *(CR*64*),* 96 *(CR*96*), and* 128 *(CR*128*). Computational times in seconds for the setup of the preconditioner (left) and for the PCG (right) are shown as a function of the number of processors.*

TABLE 2

Computational times for the main phases of the adaptive selection of constraints at the finest level in the weak scaling test for (4.1) *with random material coefficients in* $[10^{-2}, 10^2]$. *Minimum and maximum times (in seconds) are reported for the explicit computation of the Schur complement (S), its explicit inversion (S^{-1}), and the solution of all the generalized eigenvalue problems (GEP).*

procs	S		S^{-1}		GEP	
	min	max	min	max	min	max
8192	1.02	2.23	0.25	4.82	0.05	0.52
16384	0.98	2.32	0.23	5.28	0.08	0.49
32768	0.94	2.30	0.24	5.57	0.06	0.68

multilevel BDDC method are larger than in the constant coefficients case, but they are still acceptable considering the highly heterogeneous problem we are solving. The increase of setup time is due to the larger sizes of the subdomains and to the use of adaptivity at the finest levels, that in turn generate larger coarse subdomain matrices (3.11). Also, a very strict eigenvalue tolerance for the computation of primal constraints has been used at the coarser levels, in order to stress as much as possible the current `PCBDDC` code and reveal its weaknesses; as a result of the very low threshold used, coarse problem sizes at the second levels range from 12K to 141K for `CR64`, from 8K to 104K for `CR96`, and from 7K to 85K for `CR128`. The setup costs of a parallel Cholesky solver for these problems are not negligible and increase as the number of coarse subdomains increases.

Detailed timings for the three main phases of the adaptive technique employed are shown in Tables 2 and 3 for the finest and coarser levels, respectively. The phases considered are the explicit computation of the local Schur complement (S), its explicit inversion (S^{-1}), and the solution of the local generalized eigenvalue problems on subdomain faces; minimum and maximum times for each phase are reported. Results reveal that the costs for the eigenvalue computations are negligible; instead, a very poor load balancing in the local interface sizes affects the timings for the other two phases, with the explicit inversion step being most affected, since the complexity of this algorithm is cubical in the matrix size. The same issues are observed at the coarser level, and they are more pronounced with larger coarsening ratios (see Table 3). However, the performances of the adaptive multilevel BDDC algorithm for the

TABLE 3

Computational times for the main phases of the adaptive selection of constraints at the coarser levels in the weak scaling test for (4.1) *with random material coefficients in* $[10^{-2}, 10^{2}]$. *Minimum and maximum times (in seconds) are reported for the explicit computation of the Schur complement and its explicit inversion for the cases* `CR64` *and* `CR128`.

procs	S_{64}		S_{64}^{-1}		S_{128}		S_{128}^{-1}	
	min	max	min	max	min	max	min	max
8192	0.13	0.71	0.05	1.00	0.53	2.21	0.26	3.71
16384	0.18	0.82	0.07	1.31	0.56	2.32	0.31	4.41
32768	0.16	0.80	0.06	1.95	0.55	2.45	0.28	4.98

highly heterogeneous SPD problem at hand are very good in the solve phase, since the number of Krylov iterations is constant and equal to 21 (data not shown), and the solving times weakly increase as the number of subdomains increases, due to the increasing costs of the parallel Cholesky solver for the coarse problem of the coarser level.

4.3. Weak scaling at Shaheen's full scale. We end this section by reporting the results of a weak scaling test using Shaheen at full scale. The model problem considered here is the Poisson problem on the unit cube discretized with linear hexahedral elements; Dirichlet boundary conditions are imposed on a face of the unit cube, whereas Neumann conditions are imposed elsewhere on the boundary. Cubical subdomains are considered. The test is meant to validate the current multilevel implementation on a very large number of cores. The sizes of the local problems are kept fixed at 64K degrees of freedom per core, and the number of subdomains is increased to 195K, for a total of 12.4 billion degrees of freedom. Inexact subdomain solvers with nullspace corrections [25] are used applying one V-cycle of boomerAMG [35] from the HYPRE library [36], using two sweeps per multigrid level with symmetric-SOR relaxation. Vertices and edge averages have been used to customize the primal space of `PCBDDC` at the finest level; multilevel BDDC with one extra level and coarsening ratio 768 is considered for the coarse solver, using vertices constraints together with edge and face averages at the coarser level. Cardinality scaling has been used at all levels. Since ParMETIS [37] is used to the decompose the coarser meshes, the coarse subdomains no longer possess cubical shapes. Sequential Cholesky factorizations from CHOLMOD [22] are used for all the solvers at the coarser level. Figure 6 shows the timings for the setup (left panel), solve (central panel), and application (right) phases of `PCBDDC`. Setup times increase as the number of processors increases, but they re-

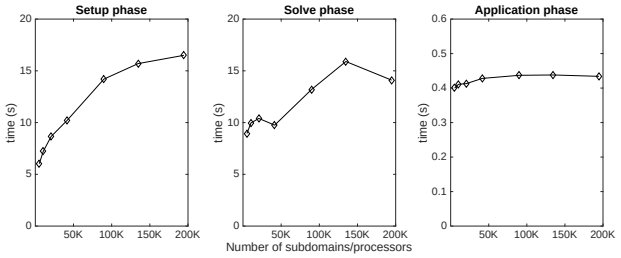

FIG. 6. *Weak scaling test for the inexact multilevel BDDC. Computational times (in seconds) for the setup of the preconditioner (left), the PCG (central), and the application of the preconditioner (right) are plotted as a function of the number of processors.*

main bounded using the machine at full scale; the increase of solving times results from the different number of iterations needed by the PCG (data not shown). Indeed, the computational times for the application of `PCBDDC` remain almost constant; the efficiency of the application stage of the preconditioner is about 92% with 195K cores, computed by using (4.2) with $p_1 = 5184$. Similar results have been obtained by other groups on an IBM BlueGene/Q machine [8].

5. Conclusions and future work. The current work has presented a novel class of preconditioners in the PETSc library which are based on BDDC techniques; the construction of the unassembled linear operators needed by the methods and the preconditioner customization have been presented. Large-scale numerical results up to 195K cores have established the scalability of the current implementation. An experimental interface to the FETI-DP method, dual of BDDC, has also been detailed.

The construction of the unassembled operator using the PETSc `MATIS` format is currently left to the user; future implementations plan to support the creation of the `MATIS` object from within the `DMPlex` container in order to natively support the most recent features of PETSc for finite element discretizations. Future implementation of the `PCBDDC` class will include a face-based selection of primal vertices [72] in order to guarantee well-posed local corrections; also, future extensions of the adaptive selection of constraints will consider different factorization packages that provide a Schur complement support, in order to remove the restriction to 32-bit integer builds which has been inherited from the external package MUMPS. An interface to `PCBDDC` from the C++ DOLFIN library is currently under development and will be integrated into the future versions of the FENICS package.

The numerical results provided in the current work for synthetic highly heterogeneous SPD systems encourage future research on the adaptive selection of constraints for symmetric indefinite or nonsymmetric linear systems. Even if it is still to be demonstrated to what extent the adaptive selection of constraints will lead to competitive BDDC algorithms in real applications, the overall technique is well suited for future exascale machines, where local flops are expected to be cheap compared to data movement [27]. Also, having the ability to control the condition number, the adaptive enrichment of the primal space makes it possible to reduce dramatically the number of iterations of the Krylov solver in a black-box fashion; this in turn reduces the number of global synchronization steps needed by the iterative solver. Sensitivity analysis on generalized eigenvalue thresholds and related strong scaling tests will be meaningful in real test cases and they will be the subject of future research. Adaptive BDDC algorithms are currently under study for the mixed formulation of porous media flows with lowest order Raviart–Thomas elements [82] and in the context of magnetic inversion [20] using lowest order Nédélec elements.

The results provided here will serve as a basis for future optimizations of the code; mesh partitioning algorithms that are able to balance the size of the local interface and not only the number of degrees of freedom per subdomains [21] are key to the success of the algorithm. Also, the possibility of exploiting hardware accelerators during the adaptive process will be the subject of future research, with a specific focus to the explicit inversion of the local Schur complements [2, 59].

Finally, future research efforts may consider the possibility of expanding the BDDC methods toward black-box hybrid solvers [1] for elliptic problems, by approximately disassembling already assembled SPD matrices in a way that preserves the fundamental requirements of the unassembled problems, even if exact information about their discrete origins is not known.

Acknowledgments. The author wishes to thank the KAUST Supercomputing Laboratory and Cray Inc. for early access to Shaheen and the Swiss National Supercomputing Centre (CSCS), in particular its director Thomas Schultess, for access to the Cray XC40 PizDora. The author is also grateful to Prof. O. B. Widlund, Prof. D. Keyes, and two anonymous referees for valuable comments and suggestions which helped improve the manuscript.

REFERENCES

[1] E. Agullo, L. Giraud, A. Guermouche, A. Haidar, and J. Roman, *Parallel algebraic domain decomposition solver for the solution of augmented systems*, Adv. Eng. Soft., 60–61 (2013), pp. 23–30.

[2] R. Al-Omairy, G. Miranda, H. Ltaief, R. M. Badia, X. Martorell, J. Labarta, and D. Keyes, *Dense matrix computations on NUMA architectures with distance-aware work stealing*, Internat. J. Supercomputing Frontiers Innovations, 1 (2015), pp. 49–72.

[3] P. R. Amestoy, I. S. Duff, J. Koster, and J. Y. L'Excellent, *A fully asynchronous multifrontal solver using distributed dynamic scheduling*, SIAM J. Matrix Anal. Appl., 23 (2001), pp. 15–41.

[4] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen, *LAPACK Users' Guide*, 3rd ed., SIAM, Philadelphia, 1999.

[5] D. N. Arnold, R. S. Falk, and J. Gopalakrishnan, *Mixed finite element approximation of the vector Laplacian with Dirichlet boundary conditions*, Math. Models Methods Appl. Sci., 22 (2012).

[6] P. Avery, G. Rebel, M. Lesoinne, and C. Farhat, *A numerically scalable dual-primal substructuring method for the solution of contact problems—Part* I*: The frictionless case*, Comput. Methods Appl. Mech. Engrg., 193 (2004), pp. 2403–2426.

[7] S. Badia, A. F. Martin, and J. Principe, *A highly scalable parallel implementation of balancing domain decomposition by constraints*, SIAM J. Sci. Comput., 36 (2014), pp. C190-C218.

[8] S. Badia, A. F. Martin, and J. Principe, *Multilevel balancing domain decomposition at extreme scales*, SIAM J. Sci. Comp., 38 (2016), pp. C22–C52.

[9] S. Balay, S. Abhyankar, M. F. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, V. Eijkhout, W. D. Gropp, D. Kaushik, M. G. Knepley, L. Curfman McInnes, K. Rupp, B. F. Smith, S. Zampini, and H. Zhang, PETSc web page, http://www.mcs.anl.gov/petsc.

[10] S. Balay, S. Abhyankar, M. F. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, V. Eijkhout, W. D. Gropp, D. Kaushik, M. G. Knepley, L. Curfman McInnes, K. Rupp, B. F. Smith, S. Zampini, and H. Zhang, *PETSc Users Manual*, ANL-95/11, Revision 3.6, 2015.

[11] L. Beirão da Veiga, C. Chinosi, C. Lovadina, and L. F. Pavarino, *Robust BDDC preconditioners for Reissner-Mindlin plate bending problems and MITC elements*, SIAM J. Numer. Anal., 47 (2010), pp. 4214–4238.

[12] L. Beirão da Veiga, C. Chinosi, C. Lovadina, and L. F. Pavarino, *BDDC preconditioners for Naghdi shell problems and MITC*9 *elements*, Comp. Struct., 102 (2012), pp. 28–41.

[13] L. Beirão da Veiga, L. F. Pavarino, S. Scacchi, O. B. Widlund, and S. Zampini, *Isogeometric BDDC preconditioners with deluxe scaling*, SIAM J. Sci. Comput., 36 (2014), pp. A1118–A1139.

[14] L. Beirão da Veiga, L. F. Pavarino, S. Scacchi, O. B. Widlund, and S. Zampini, *BDDC deluxe for isogeometric analysis*, in Proceedings of the 22nd International Conference on Domain Decomposition Methods in Science and Engineering, 2014, pp. 15–28.

[15] L. Beirão da Veiga, L. F. Pavarino, S. Scacchi, O. B. Widlund, and S. Zampini, *Adaptive Selection of Primal Constraints for Isogeometric BDDC Deluxe Preconditioners*, Technical report TR2015-977, Courant Institute, New York, 2015.

[16] F. Brezzi and M. Fortin, *Mixed and Hybrid Finite Element Methods*, Springer, New York, 1991.

[17] Z. Cai, R. D. Lazarov, T. A. Manteuffel, and S. F. McCormick, *First-order system least squares for second-order partial differential equations: Part* II, SIAM J. Numer. Anal., 34 (1994), pp. 425–454.

[18] J. G. Calvo, *A BDDC algorithm with deluxe scaling for H(curl) in two dimensions for irregular subdomains*, Math. Comp., 85 (2016), pp. 1085–1111.

[19] J. G. Calvo and O. B. Widlund, *An Adaptive Choice of Primal Constraints for BDDC Domain Decomposition Algorithms*, Technical report TR2015-979, Courant Institute, New York 2015.
[20] J. M. Carcione, *Simulation of electromagnetic diffusion in anisotropic media*, Progr. Electromagnet. Res. B, 26 (2010), pp. 425–450.
[21] A. Casadei, P. Ramet, and J. Roman, *An improved recursive graph bipartitioning algorithm for well balanced domain decomposition*, in Proceedings of the 22nd Annual IEEE International Conference on High Performance Computing (HiPC), 2014.
[22] Y. Chen, T. A. Davis, W. W. Hager, and S. Rajamanickam, *Algorithm* 887*: CHOLMOD, supernodal sparse factorization and update/downdate*, AMC Trans. Math. Software, 35 (2008), pp. 1–14.
[23] J. M. Cros, *A preconditioner for the Schur complement domain decomposition method*, in Proceedings of the 14th International Conference on Domain Decomposition Methods in Science and Engineering, 2003, pp. 373–380.
[24] C. R. Dohrmann, *A preconditioner for substructuring based on constrained energy minimization*, SIAM J. Sci. Comput., 25 (2003), pp. 246–258.
[25] C. R. Dohrmann, *An approximate BDDC preconditioner*, Numer. Linear Algebra Appl., 14 (2007), pp. 149–168.
[26] C. R. Dohrmann and O. B. Widlund, *A BDDC algorithm with deluxe scaling for three-dimensional H(curl) problems*, Comm. Pure Appl. Math., 69 (2015).
[27] J. Dongarra, P. Beckman, et al., *The international exascale software project roadmap*, Int. J. High Perf. Comp. Appl., 6 (2011), pp. 1–58.
[28] M. Dryja, J. Galvis, and M. Sarkis, *BDDC methods for discontinuous Galerkin discretization of elliptic problems*, J. Complexity, 23 (2007), pp. 715–739.
[29] C. Farhat, M. Lesoinne, P. Le Tallec, 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.
[30] C. Farhat and J. Li, *An iterative domain decomposition method for the solution of a class of indefinite problems in computational structural dynamics*, Appl. Numer. Math., 54 (2005), pp. 150–166.
[31] C. Farhat, J. Li, and P. Avery, *A FETI-DP method for the parallel iterative solution of indefinite and complex-valued solid and shell vibration problems*, Internat. J. Numer. Methods Engrg., 63 (2005), pp. 398–427.
[32] Y. Fragakis and M. Papadrakakis, *The mosaic of high performance domain decomposition methods for structural mechanics: Formulation, interrelation and numerical efficiency of primal and dual methods*, Comput. Methods Appl. Mech. Engrg., 192 (2003), pp. 3799–3830.
[33] J. Galvis and Y. Efendiev, *Domain decomposition preconditioners for multiscale flows in high-contrast media*, Multiscale Model. Simul., 8 (2010), pp. 1461–1483.
[34] B. Hadri, S. Kortas, S. Feki, R. Khurram, and G. Newby *Overview of the KAUST's Cray X*40 *System: Shaheen* II, presented at Cray User Group Conference, Chicago, 2015.
[35] V. E. Henson and U. M. Yang, BoomerAMG: A parallel algebraic multigrid solver and preconditioner, Appl. Numer. Math., 41 (2002), pp. 155–177.
[36] *HYPRE: High Performance Preconditioners*, http://www.llnl.gov/CASC/hypre/.
[37] G. Karypis, *METIS and ParMETIS*, in Encyclopedia of Parallel Computing, Springer, New York, 2011, pp. 1117–1124.
[38] H. H. Kim, *A BDDC algorithm for mortar discretization of elasticity problems*, SIAM J. Numer. Anal., 46 (2008), pp. 2090–2111.
[39] H. H. Kim, *A FETI-DP formulation of three-dimensional elasticity problems with mortar discretization*, SIAM J. Numer. Anal., 46 (2008), pp. 2346–2370.
[40] H. H. Kim, C. O. Lee, and E. H. Park, *A FETI-DP formulation for the Stokes problems without primal pressure components*, SIAM J. Numer. Anal., 47 (2010), pp. 4142–4162.
[41] H. H. Kim and E. T. Chung, *A BDDC algorithm with optimally enriched coarse spaces for two-dimensional elliptic problems with oscillatory and high contrast coefficients*, SIAM J. Multiscale Model. Simul., 13 (2015), pp. 571–593.
[42] H. H. Kim, E. T. Chung, and J. Wang, *BDDC and FETI-DP Algorithms with Adaptive Coarse Spaces for Three-Dimensional Elliptic Problems with Oscillatory and High Contrast Coefficients*, submitted.
[43] A. Klawonn, M. Lanser, and O. Rheinbach, *Nonlinear FETI-DP and BDDC Methods*, SIAM J. Sci. Comput., 36 (2014), pp. A737–A765.
[44] A. Klawonn, M. Kühn, and O. Rheinbach, *Adaptive Coarse Spaces for FETI-DP in Three Dimensions*, TR 11/1025, Freiberg University, 2015.

[45] A. Klawonn, P. Radtke, and O. Rheinbach, *FETI-DP methods with an adaptive coarse space*, SIAM J. Numer. Anal., 53 (2015), pp. 297–320.
[46] A. Klawonn, P. Radtke, and O. Rheinbach, *A comparison of adaptive coarse spaces for substructuring in two dimensions*, Electron. Trans. Numer. Anal., 45 (2016), pp. 75–106.
[47] A. Klawonn and O. Rheinbach, *Deflation, projector preconditioning, and balancing in iterative substructuring methods: connections and new results*, SIAM J. Sci. Comput., 34 (2012), pp. A459–A484.
[48] A. Klawonn and O. Rheinbach, *Robust FETI-DP methods for heterogeneous three dimensional elasticity problems*, Comput. Methods Appl. Mech. Engrg., 196 (2007), pp. 1400–1414.
[49] A. Klawonn and O. Rheinbach, *Highly scalable parallel domain decomposition methods with an application to biomechanics*, ZAMM Z. Angew. Math. Mech., 90 (2010), pp. 5–32.
[50] A. Klawonn, O. Rheinbach, and O. B. Widlund, *An analysis of a FETI-DP algorithm on irregular subdomains in the plane*, SIAM J. Numer. Anal., 46 (2008), pp. 2484–2504.
[51] A. Klawonn and O. B. Widlund, *Dual-primal FETI methods for linear elasticty*, Comm. Pure Appl. Math., 59 (2006), pp. 1523–1572.
[52] T. V. Kolev and P. S. Vassilevski, *Parallel auxiliary space AMG solver for H(div) problems*, SIAM J. Sci. Comput., 34 (2012), pp. A3079–A3098.
[53] J. H. Lee, *A balancing domain decomposition by constraints deluxe method for Reissner–Mindlin plates with Falk–Tu elements*, SIAM J. Numer. Anal., 53 (2015), pp. 63–81.
[54] J. Li and O. B. Widlund, *FETI-DP, BDDC, and block Cholesky methods*, Internat. J. Numer. Methods Engrg., 66 (2006), pp. 250–271.
[55] J. Li and O. B. Widlund, *BDDC algorithms for incompressible Stokes equations*, SIAM J. Numer. Anal., 44 (2006), pp. 2432–2455.
[56] P. Lin, *A sequential regularization method for time-dependent incompressible Navier-Stokes equations*, SIAM J. Numer. Anal., 34 (1997), pp. 105–1071.
[57] A. Logg, K.-A. Mardal, G. N. Wells, et al., *Automated Solution of Differential Equations by the Finite Element Method*, Lect. Notes Comput. Sci. Eng. 84, Springer, New York, 2010.
[58] A. Logg and G. N. Wells, *DOLFIN: Automated finite element computing*, ACM Trans. Math. Software, 37 (2012).
[59] H. Ltaief, S. Tomov, R. Nath, P. Du, and J., Dongarra, *A scalable high performant Cholesky factorization for multicore with GPU accelerators*, in Proceedings of the High Performance Computing for Computational Science (VECPAR), 2011, pp. 93–101.
[60] J. Mandel, *Balancing domain decomposition*, Comm. Appl. Numer. Methods, 9 (1993), pp. 233–241.
[61] J. Mandel and C. R. Dohrmann, *Convergence of a balancing domain decomposition by constraints and energy minimization*, Numer. Linear Algebra Appl., 10 (2003), pp. 639–659.
[62] J. Mandel, C. R. Dohrmann, and R. Tezaur, *An algebraic theory for primal and dual substructuring methods by constraints*, Appl. Numer. Math., 54 (2005), pp. 167–193.
[63] J. Mandel and B. Sousedík, *Adaptive selection of face coarse degrees of freedom in the BDDC and the FETI-DP iterative substructuring methods*, Comput. Methods Appl. Mech. Engrg., 196 (2007), pp. 1389–1399.
[64] T. Mathew, *Domain Decomposition Methods for the Numerical Solution of Partial Differential Equations*, Lect. Notes Comput. Sci. Eng. 61, Springer, New York, 2008.
[65] D.-S. Oh, O. B. Widlund, S. Zampini, and C. R. Dohrmann, *BDDC Algorithms with Deluxe Scaling and Adaptive Selection of Primal Constraints for Raviart-Thomes Vector Fields*, Technical report TR2015-978, Courant Institute, 2015.
[66] L. F. Pavarino, *BDDC and FETI-DP preconditioners for spectral element discretizations*, Comput. Methods Appl. Mech. Eng., 196 (2007), pp. 1380–1388.
[67] L. F. Pavarino, O. B. Widlund, and S. Zampini, *BDDC preconditioners for spectral element discretizations of almost incompressible elasticity in three dimensions*, SIAM J. Sci. Comput., 32 (2010), pp. 3604–3626.
[68] L. F. Pavarino, S. Scacchi, and S. Zampini, *Newton-Krylov-BDDC solvers for nonlinear cardiac mechanics*, Comp. Methods Appl. Mech. Eng., 295 (2015), pp. 562–580.
[69] C. Pechstein and C. R. Dohrmann, *Modern Domain Decomposition Methods BDDC, Deluxe Scaling, and an Algebraic Approach*, http://people.ricam.oeaw.ac.at/c.pechstein/pechstein-bddc2013.pdf (2013).
[70] C. Pechstein and R. Scheichl, *Analysis of FETI methods for multiscale PDEs*, Numer. Math., 111 (2008), pp. 293–333.
[71] C. Pechstein, M. Sarkis, and R. Scheichl, *New theoretical coefficient robustness for FETI-DP*, in Domain Decomposition Methods in Science and Engineering 20, R. Bank, M. Holst, O. Widlund, and J. Xu, eds., Lect. Notes Comput. Sci. Eng. 91, Springer, New York, 2013, pp. 313–320.

[72] J. Šístek, M. Čertíková, P. Burda, and J. Novotný, *Face-based selection of corners in 3D substructuring*, Math. Comput. Simulation, 82 (2012), pp. 1799–1811.
[73] J. Šístek, J. Brezina, and B. Sousedík, *BDDC for Mixed-Hybrid Formulation of Flow in Porous Media with Combined Mesh Dimensions*, arXiv:1504.07085, 2015.
[74] B. F. Smith, P. Bjørstad, and W. Gropp, *Domain Decomposition: Parallel Multilevel Methods for Elliptic Differential Equations*, Cambridge University Press, Cambridge, UK, 1996.
[75] B. Sousedík, J. Šístek, and J. Mandel, *Adaptive-multilevel BDDC and its parallel implementation*, Computing, 95 (2013), pp. 1087–1119.
[76] N. Spillane, V. Dolean, P. Hauret, F. Nataf, and D. J. Rixen, *Solving generalized eigenvalue problems on the interfaces to build a robust two-level FETI method*, C. R. Math. Acad. Sci. Paris, 351 (2013), pp. 197–201.
[77] N. Spillane and D. J. Rixen, *Automatic spectral coarse spaces for robust finite element tearing and interconnecting and balanced domain decomposition algorithms*, Internat. J. Numer. Methods Engrg., 95 (2013), pp. 953–990.
[78] N. Spillane, V. Dolean, P. Hauret, F. Nataf, C. Pechstein, and R. Scheichl, *Abstract robust coarse spaces for systems of PDEs via gereralized eigenproblems in the overlap.*, Numer. Math., 126 (2014), pp. 741–770.
[79] A. Toselli and O. B. Widlund, *Domain Decomposition Methods: Algorithms and Theory*, Springer-Verlag, Berlin, 2005.
[80] X. Tu, *Three-level BDDC in three dimensions*, SIAM J. Sci. Comput., 29 (2007), pp. 1759–1780.
[81] X. Tu, *Three-level BDDC in two dimensions*, Internat. J. Numer. Methods Engrg., 69 (2007), pp. 33–59.
[82] X. Tu, *A BDDC algorithm for a mixed formulation of flow in porous media*, Electron. Trans. Numer. Anal., 20 (2005), pp. 164–179.
[83] X. Tu, *A BDDC algorithm for flow in porous media with a hybrid finite element discretization*, Electron. Trans. Numer. Anal., 26 (2007), pp. 146–160.
[84] X. Tu and J. Li, *A balancing domain decomposition method by constraints for advection-diffusion problems*, Commun. Appl. Math. Comput. Sci., 3 (2008), pp. 25–60.
[85] P. S. Vassilevski and U. Villa, *A block-diagonal algebraic multigrid preconditioner for the Brinkman problem*, SIAM J. Sci. Comput., 35 (2013), pp. S3–S17.
[86] O. B. Widlund and C. R. Dohrmann, *BDDC deluxe domain decomposition*, in Proceedings of the 22nd International Conference on Domain Decomposition Methods in Science and Engineering, 2014.
[87] S. Zampini, *Inexact BDDC methods for the cardiac bidomain model*, in Proceedings of the 21st International Conference on Domain Decomposition Methods in Science and Engineering, 2012, pp. 247–255
[88] S. Zampini, *Dual-primal methods for the cardiac bidomain model*, Math. Models Methods Appl. Sci., 24 (2014), pp. 667–696.