Electronic Transactions on Numerical Analysis.
Volume 49, pp. 244–273, 2018.

ISSN 1068–9613.
DOI: 10.1553/etna_vol49s244

ETNA
Kent State University and
Johann Radon Institute (RICAM)

NONLINEAR BDDC METHODS WITH APPROXIMATE SOLVERS*

AXEL KLAWONN[†‡], MARTIN LANSER[†‡], AND OLIVER RHEINBACH[§]

Abstract. New nonlinear BDDC (Balancing Domain Decomposition by Constraints) domain decomposition methods using inexact solvers for the subdomains and the coarse problem are proposed. In nonlinear domain decomposition methods, the nonlinear problem is decomposed before linearization to improve concurrency and robustness. For linear problems, the new methods are equivalent to known inexact BDDC methods. The new approaches are therefore discussed in the context of other known inexact BDDC methods for linear problems. Relations are pointed out, and the advantages of the approaches chosen here are highlighted. For the new approaches, using an algebraic multigrid method as a building block, parallel scalability is shown for more than half a million (524 288) MPI ranks on the JUQUEEN IBM BG/Q supercomputer (JSC Jülich, Germany) and on up to 193 600 cores of the Theta Xeon Phi supercomputer (ALCF, Argonne National Laboratory, USA), which is based on the recent Intel Knights Landing (KNL) many-core architecture. One of our nonlinear inexact BDDC domain decomposition methods is also applied to three-dimensional plasticity problems. Comparisons to standard Newton-Krylov-BDDC methods are provided.

Key words. nonlinear BDDC, nonlinear domain decomposition, nonlinear elimination, Newton's method, nonlinear problems, parallel computing, inexact BDDC, nonlinear elasticity, plasticity

AMS subject classifications. 68W10, 68U20, 65N55, 65F08, 65Y05

1. Introduction.

Nonlinear BDDC (Balancing Domain Decomposition by Constraints) domain decomposition methods were introduced in [24] for the parallel solution of nonlinear problems and can be viewed as generalizations of the well-known family of BDDC methods for linear elliptic problems [11, 15, 35, 37]. They can also be seen as nonlinearly right-preconditioned Newton methods [30]. Classical, linear BDDC methods are closely related to the earlier linear FETI-DP (Finite Element Tearing and Interconnecting-Dual Primal) methods, e.g., [18, 33, 41]. The connection of nonlinear FETI-DP and nonlinear BDDC methods is weaker [24].

In our nonlinear BDDC method with inexact solvers, the nonlinear problem

$$A(\bar{u}) = 0$$

is first reduced to the interface Γ by a nonlinear elimination of all degrees of freedom in the interior of the subdomains. This results in a nonlinear Schur complement operator (see (2.9))

$$S_\Gamma(u_\Gamma),$$

defined on the interface. After a Newton linearization, the tangent $DS_{\Gamma\Gamma}$ of the nonlinear Schur complement S_Γ (see (2.16)) can then be preconditioned by a BDDC preconditioner; cf. [24, Sections 4.2 and 4.3]. As opposed to [24], however, in this paper we will return to an equivalent formulation using the tangent DA of the original operator A in order to apply a BDDC method with inexact subdomain solvers. Moreover, we will write our nonlinear BDDC method in the form of a nonlinearly right-preconditioned Newton method; see Section 2.2.

*Received July 31, 2017. Accepted October 12, 2018. Published online on December 19, 2018. Recommended by Martin J. Gander. This work was supported in part by Deutsche Forschungsgemeinschaft (DFG) through the Priority Programme 1648 "Software for Exascale Computing" (SPPEXA) under grants KL 2094/4-1, KL 2094/4-2, RH 122/2-1, and RH 122/3-2.

[†]Mathematical Institute, University of Cologne, Weyertal 86-90, 50931 Köln, Germany
({axel.klawonn, martin.lanser}@uni-koeln.de).

[‡]Center for Data and Simulation Science, University of Cologne, Germany.

[§]Institut für Numerische Mathematik und Optimierung, Fakultät für Mathematik und Informatik, Technische Universität Bergakademie Freiberg, Akademiestr. 6, 09596 Freiberg
(oliver.rheinbach@math.tu-freiberg.de).

ETNA
Kent State University and
Johann Radon Institute (RICAM)

Nonlinear BDDC and FETI-DP methods are related to nonlinear FETI-1 methods [39] and nonlinear Neumann-Neumann methods [6]. Other nonlinear domain decomposition (DD) methods are the ASPIN (Additive Schwarz Preconditioned Inexact Newton) method [9, 10, 19, 20, 22, 23] and RASPEN (Restricted Additive Schwarz Preconditioned Exact Newton) [17].

In the nonlinear DD methods in [24, 30], exact solvers are used to solve the linearized local subdomain problems and the coarse problem. As for linear DD methods, this can limit the parallel scalability. For linear problems, these limits have been overcome by using efficient preconditioners for the coarse problem [32, 38, 42] or the local subdomain problems [16, 32, 34]. The currently largest range of weak parallel scalability of a domain decomposition method (to almost 800 000 processor cores) was achieved for elasticity in [27] by a combination of a nonlinear FETI-DP method [24] with an inexact solver for the coarse problem [26, 32] using the complete Mira supercomputer at Argonne National Laboratory (USA). Similar results were achieved for a BDDC method for scalar elliptic problems using an inexact solver for the coarse problem [2].

The present paper extends the nonlinear BDDC methods from [24, 30] by a combination of different inexact BDDC approaches. For linear problems, the new inexact methods are, depending on the choice of the linear preconditioner, equivalent to one of the inexact BDDC methods introduced in [36] and related to the preconditioners introduced in [16].

One nonlinear BDDC variant is closely related to a recent nonlinear FETI-DP method with inexact solvers (proposed in [28]), which has already been tested for more than half a million MPI ranks on the JUQUEEN supercomputer (Jülich Supercomputing Centre, Germany). At the core of both of these methods, i.e., the one proposed here and the nonlinear FETI-DP method [28], is the choice of an efficient parallel preconditioner for the $D\widetilde{K}$ operator; see (2.20) in Section 2.3. For this, we will choose a parallel AMG (algebraic multigrid) method [21], which has scaled to more than half a million MPI ranks for linear elasticity by itself [3].

2. Nonlinear BDDC.

In this section, we first present the nonlinear BDDC method as introduced in [24] and then derive our new nonlinear BDDC method with inexact solvers.

2.1. A nonlinear finite element problem.

Let $\Omega \subset \mathbb{R}^d$, $d = 2, 3$, be a computational domain and Ω_i, $i = 1, \ldots, N$, a nonoverlapping domain decomposition such that $\Omega = \bigcup_{i=1}^{N} \Omega_i$. Each subdomain Ω_i is discretized by finite elements, the corresponding local finite element spaces are denoted by W_i, $i = 1, \ldots, N$, and the product space is defined by $W = W_1 \times \cdots \times W_N$. We further define the global finite element space corresponding to Ω by $V^h = V^h(\Omega)$ and the isomorphic space $\widehat{W} \subset W$ of all functions from W which are continuous in all interface variables between the subdomains. The interface can be written as $\Gamma := \bigcup_{i=1}^{N} \partial\Omega_i \setminus \partial\Omega$. We are interested in solving a nonlinear problem

$$A(\bar{u}) = 0 \tag{2.1}$$

in the global finite element space V^h. In our context, it is useful to reformulate problem (2.1) using local nonlinear finite element problems

$$K_i(u_i) = f_i$$

in the local subdomain finite element spaces W_i, $i = 1, \ldots, N$. Here, we have $u_i \in W_i$, $K_i : W_i \to W_i$, and the right-hand sides f_i contain all terms that are independent of u_i, e.g., static volume forces. These local problems originate from the finite element discretization of a nonlinear partial differential equation on the subdomains. The explicit form of $K_i(u_i)$ and f_i is strongly problem dependent. A detailed description how to obtain $K_i(u_i)$ and f_i for a given

ETNA
Kent State University and
Johann Radon Institute (RICAM)

TABLE 2.1
Properties of a good nonlinear preconditioner N.

1. $N : V^h \to U \subset V^h$,
2. the solution g^* of $A(g^*) = 0$ is in the range of N,
3. N puts the current iterate into the neighborhood of the solution; see also [8], and
4. $N(\bar{u})$ is easily computable compared to the inverse action of $A(\bar{u})$.

nonlinear differential equation is presented in [24, Section 5.1]. There, we use the p-Laplace equation as a model problem.

Using restriction operators $R_i : V^h \to W_i$, $i = 1, \dots N$, and the notation

$$K(u) := \begin{bmatrix} K_1(u_1) \\ \vdots \\ K_N(u_N) \end{bmatrix}, \quad f := \begin{bmatrix} f_1 \\ \vdots \\ f_N \end{bmatrix}, \quad u := \begin{bmatrix} u_1 \\ \vdots \\ u_N \end{bmatrix}, \quad R := \begin{bmatrix} R_1 \\ \vdots \\ R_N \end{bmatrix},$$

we can write (2.1) in the form

$$\underbrace{R^T K(R\bar{u}) - R^T f}_{=A(\bar{u})} = 0. \tag{2.2}$$

Using (2.2) and the chain rule, we can assemble the Jacobian matrix $DA(\bar{u})$ of $A(\bar{u})$ using the local tangential matrices $DK_i(R_i\bar{u})$, $i = 1, \dots, N$, i.e.,

$$DA(\bar{u}) = R^T \; DK(R\bar{u}) \; R, \tag{2.3}$$

where

$$DK(R\bar{u}) = \begin{bmatrix} DK_1(R_1\bar{u}) & & \\ & \ddots & \\ & & DK_N(R_N\bar{u}) \end{bmatrix}$$

is the block-diagonal matrix of the local subdomain tangential matrices.

2.2. Nonlinear BDDC methods. After having introduced a nonlinear problem and some useful notation, we now define a nonlinear BDDC algorithm.

In the standard Newton-Krylov-BDDC method, problem (2.1) is first linearized and then solved by a Krylov subspace method using a BDDC domain decomposition preconditioner constructed for the tangent matrix. Instead, we consider the decomposed nonlinear problem (2.2) and insert a nonlinear right preconditioner, which represents the nonlinear elimination of subdomain interior variables.

We have recently introduced a unified framework which allows to compress all nonlinear FETI-DP and BDDC methods into a single algorithm; see [30] for a more detailed description of nonlinear preconditioning in FETI-DP and BDDC methods. We briefly summarize the general ideas from [30] for better readability. Instead of solving (2.2) by Newton's method, we solve the nonlinearly preconditioned system

$$A(N(\bar{u})) = 0, \tag{2.4}$$

where N is a nonlinear preconditioner fulfilling the properties shown in Table 2.1.

Applying Newton's method to (2.4) and applying a BDDC preconditioner to the tangent leads to the nonlinear BDDC algorithm presented in Algorithm 1.

ETNA
Kent State University and
Johann Radon Institute (RICAM)

Algorithm 1: Nonlinear BDDC algorithm; taken from [30].

Init: $\bar{u}^{(0)}$
Iterate over k**:**
  Compute: $g^{(k)} := N(\bar{u}^{(k)})$
  If $||A(g^{(k)})||$ sufficiently small
    break; /* Convergence of nonlinear right-preconditioned method */
  Solve iteratively with some linear BDDC preconditioner:
  $DA(g^{(k)})DN(\bar{u}^{(k)})\ \delta\bar{u}^{(k)} = A(g^{(k)})$
  Update: $\bar{u}^{(k+1)} := \bar{u}^{(k)} - \alpha^{(k)}\delta\bar{u}^{(k)}$
End Iteration

Let us remark that the choice of the nonlinear preconditioner N and the linear BDDC preconditioner to solve the linearized system $DA(g^{(k)})DN(\bar{u}^{(k)})\ \delta\bar{u}^{(k)} = A(g^{(k)})$ in Algorithm 1 are independent of each other and thus will be discussed separately. As with all right-preconditioned methods, we are searching for a solution $g^* = N(\bar{u}^*)$ of $A(g^*) = 0$ but are technically not interested in $\bar{u}^*$. The existence of g^* has to be guaranteed, which requires Assumption 2 in Table 2.1 to be fulfilled.

2.3. Choosing a nonlinear preconditioner. We will now discuss two different choices for the nonlinear right preconditioner N.

The Newton-Krylov method. The trivial choice

$$N(\bar{u}) = N_{NK}(\bar{u}) := \bar{u}$$

reduces to the well-known Newton-Krylov-BDDC (NK-BDDC) approach. The NK-BDDC method will be used as a baseline to which we can compare our new parallel implementation of the nonlinear BDDC method (NL-BDDC) with inexact solvers.

The nonlinear Schur complement method. As in [24], a nonlinear elimination of all degrees of freedom in the interior of the subdomains is performed before linearization. To derive NL-BDDC within our framework, we first have to decompose and sort all degrees of freedom in the interior part of all subdomains (I) and the interface (Γ), which is also common in linear domain decomposition methods. We obtain

$$A(\bar{u}) = \begin{bmatrix} A_I(\bar{u}) \\ A_\Gamma(\bar{u}) \end{bmatrix} = \begin{bmatrix} K_I(R\bar{u}) - f_I \\ R_\Gamma^T K_\Gamma(R\bar{u}) - R_\Gamma^T f_\Gamma \end{bmatrix} = R^T K(R\bar{u}) - R^T f, \tag{2.5}$$

where

$$R^T = \begin{bmatrix} I & 0 \\ 0 & R_\Gamma^T \end{bmatrix}.$$

For the tangent $DA(\bar{u})$ (see (2.3)) we have the following block form of the tangential matrix

$$DA(\bar{u}) = \begin{bmatrix} DK_{II}(R\bar{u}) & DK_{I\Gamma}(R\bar{u})R_\Gamma \\ R_\Gamma^T DK_{\Gamma I}(R\bar{u}) & R_\Gamma^T DK_{\Gamma\Gamma}(R\bar{u})R_\Gamma \end{bmatrix}. \tag{2.6}$$

We now define the second nonlinear preconditioner implicitly by the first line of (2.5), i.e.,

$$K_I(N_{NL,I}(\bar{u}), R_\Gamma\bar{u}_\Gamma) - f_I = 0, \tag{2.7}$$

ETNA
Kent State University and
Johann Radon Institute (RICAM)

where $\bar{u} = (\bar{u}_I^T, \bar{u}_\Gamma^T)^T$ and N_{NL} is then defined as

$$N(\bar{u}) = N_{NL}(\bar{u}) := (N_{NL,I}(\bar{u})^T, \bar{u}_\Gamma^T)^T. \tag{2.8}$$

It follows from (2.7) that the nonlinear right preconditioner N_{NL} removes the nonlinear residual in the interior of the subdomains. It represents the nonlinear elimination of the subdomain interior variables $\bar{u}_I$. The nonlinear operator

$$\begin{bmatrix} 0 \\ S_{\Gamma\Gamma}(\bar{u}_\Gamma) \end{bmatrix} = A(N_{NL}(\bar{u})) \tag{2.9}$$

represents the nonlinear Schur complement on the interface Γ; see [24, eq. (4.11)].

Details of the nonlinear Schur complement method. In each step the computation of $g^{(k)} := N_{NL}(\bar{u}^{(k)})$ in Algorithm 1 is performed using an inner Newton iteration

$$g_{I,l+1}^{(k)} := g_{I,l}^{(k)} + \delta g_{I,l}^{(k)},$$

converging to $g_I^{(k)}$, with

$$\delta g_{I,l}^{(k)} := -\left(DK(g_{I,l}^{(k)})_{II}\right)^{-1} \left(K_I(g_{I,l}^{(k)}, R_\Gamma \bar{u}_\Gamma^{(k)}) - f_I\right), \tag{2.10}$$

and $g^{(k)} = (g_I^{(k)}, \bar{u}_\Gamma^{(k)})$. Since (2.7) exclusively operates on interior degrees of freedom, the inner Newton iteration can be carried out on each subdomain individually and thus in parallel.

We will now take a closer look at the linear system

$$DA(g^{(k)}) DN_{NL}(\bar{u}^{(k)})\, \delta\bar{u}^{(k)} = A(g^{(k)}), \tag{2.11}$$

which is obtained from a Newton linearization of the right-preconditioned nonlinear system (2.4). Using (2.6), (2.7), and the chain rule, it can be easily verified that

$$DA(g^{(k)}) = \begin{bmatrix} DK_{II}(Rg^{(k)}) & DK_{I\Gamma}(Rg^{(k)})R_\Gamma \\ R_\Gamma^T DK_{\Gamma I}(Rg^{(k)}) & R_\Gamma^T DK_{\Gamma\Gamma}(Rg^{(k)})R_\Gamma \end{bmatrix}, \tag{2.12}$$

$$DN_{NL}(\bar{u}^{(k)}) = \begin{bmatrix} 0 & -DK_{II}^{-1}(Rg^{(k)})DK_{I\Gamma}(Rg^{(k)})R_\Gamma \\ 0 & I \end{bmatrix}, \tag{2.13}$$

and

$$A(g^{(k)}) = \begin{bmatrix} 0 \\ R_\Gamma^T K_\Gamma(Rg^{(k)}) - R_\Gamma^T f_\Gamma \end{bmatrix}. \tag{2.14}$$

Using $\delta\bar{u}^{(k)} = (\delta\bar{u}_I^{(k)}, \delta\bar{u}_\Gamma^{(k)})$, (2.12), (2.13), and (2.14), our equation (2.11) becomes

$$\begin{bmatrix} 0 & 0 \\ 0 & DS_{\Gamma\Gamma}(g^{(k)}) \end{bmatrix} \begin{bmatrix} \delta\bar{u}_I^{(k)} \\ \delta\bar{u}_\Gamma^{(k)} \end{bmatrix} = \begin{bmatrix} 0 \\ R_\Gamma^T K_\Gamma(Rg^{(k)}) - R_\Gamma^T f_\Gamma \end{bmatrix}, \tag{2.15}$$

where

$$\begin{aligned} &DS_{\Gamma\Gamma}(g^{(k)}) := \\ &R_\Gamma^T DK_{\Gamma\Gamma}(Rg^{(k)})R_\Gamma - R_\Gamma^T DK_{\Gamma I}(Rg^{(k)}) \left(DK_{II}(Rg^{(k)})\right)^{-1} DK_{I\Gamma}(Rg^{(k)})R_\Gamma \end{aligned} \tag{2.16}$$

ETNA
Kent State University and
Johann Radon Institute (RICAM)

is the Schur complement of the tangential matrix on the interface Γ. The operator $DS_{\Gamma\Gamma}$ can also be viewed, by the implicit function theorem, as the tangent of the nonlinear Schur complement in (2.9); see [24]. A solution of (2.15) can be obtained by solving the Schur complement system

$$DS_{\Gamma\Gamma}(g^{(k)})\,\delta\bar{u}_\Gamma^{(k)} = R_\Gamma^T K_\Gamma(Rg^{(k)}) - R_\Gamma^T f_\Gamma.$$

Since $\bar{u}_I$ has been eliminated, we can choose an arbitrary value for $\delta\bar{u}_I^{(k)}$. Indeed, $\delta\bar{u}_I^{(k)}$ is only used as an update for the initial value for the computation of $g_I^{(k+1)}$. Let us remark that we obtain the original nonlinear BDDC method as suggested in [24] by choosing $\delta\bar{u}_I^{(k)} = 0$.

A formulation allowing the use of an inexact BDDC method. In this paper, we decide to use a slightly different approach and return to the full system (2.11) instead of the Schur complement system (2.15). To be able to use inexact linear BDDC preconditioners, which have to be applied to the completely assembled tangential matrix $DA(\cdot)$, we remove the inner derivative $DN_{NL}(\cdot)$ in the system (2.11) and simply solve

$$DA(g^{(k)})\,\delta\bar{u}^{(k)} = A(g^{(k)}) \tag{2.17}$$

instead. It can easily be seen that, because of the identity block in (2.13), this does not change the update $\delta\bar{u}_\Gamma^{(k)}$ of the interface variables. The Newton step is thus not changed using this modification. However, by solving (2.17), we also compute a $\delta\bar{u}_I^{(k)}$ in the interior of the subdomains. We use this to improve the initial value for the computation of $g^{(k+1)}$ by performing the optional update $g_I^{(k)} + \delta\bar{u}_I^{(k)}$. We have found that this strategy can lead to significant improvements in practice, i.e., less inner Newton iterations and improved robustness. Omitting the interior update is, in our experience, never beneficial and therefore we will not discuss this in detail in this paper. Nonetheless, we give a brief example in Figure 4.9.

2.4. Choosing a linear BDDC preconditioner. In this section, we discuss how to precondition the linearized system $DA(g^{(k)})\,\delta\bar{u}^{(k)} = A(g^{(k)})$ (see (2.17)), which is solved by a Krylov subspace method. We consider the version of the BDDC method which is applied to the assembled operator $DA(g^{(k)})$, whereas the original BDDC preconditioner [11, 15, 35, 37] is applied to the Schur complement $DS_{\Gamma\Gamma}(g^{(k)})$.

For the construction of linear BDDC preconditioners applicable to the assembled linear system $DA(g^{(k)}) = R^T DK(Rg^{(k)})R$, we have to further subdivide the interface Γ into primal (Π) and the remaining dual (Δ) degrees of freedom. Partial finite element assembly in the primal variables is used to enforces continuity in these degrees of freedom. The primal variables usually represent vertices of subdomains but can also represent edge or face averages after a change of basis; see, e.g., [24, 35] and the references therein. Let us therefore introduce the space $\widetilde{W} \subset W$ of functions which are continuous in all primal variables as well as the corresponding restriction operator $\overline{R} : \widetilde{W} \to W$. The partially assembled tangential system is then obtained in the usual fashion by

$$D\widetilde{K}(g^{(k)}) := \overline{R}^T DK(Rg^{(k)})\overline{R}.$$

With a scaled operator $\widetilde{R}_D : V^h \to \widetilde{W}$, we can define a first BDDC-type preconditioner by

$$M_1^{-1}(g^{(k)}) := \widetilde{R}_D^T \left(D\widetilde{K}(g^{(k)})\right)^{-1} \widetilde{R}_D \tag{2.18}$$

as suggested in [36, p. 1417]. The operator $\widetilde{R}_D$ is obtained from the restriction $\widetilde{R} : V^h \to \widetilde{W}$ by multiplying each row of $\widetilde{R}$, which corresponds to a dual variable, by the inverse of

ETNA
Kent State University and
Johann Radon Institute (RICAM)

the multiplicity, i.e., the number of subdomains which the corresponding physical node belongs to. The linear preconditioner M_1^{-1} can be interpreted as a lumped version of the standard BDDC preconditioner since, following [36, Theorem 2], the preconditioned system $M_1^{-1}(g^{(k)})DA(g^{(k)})$ has, except for some eigenvalues equal to 0 and 1, the same eigenvalues as the preconditioned FETI-DP system with a lumped preconditioner. This preconditioner reduces the computational cost per iteration but also results in a linear factor of H_s/h in the condition number estimate and thus often in slow convergence. It should only be used when memory is scarce.

To obtain a more robust BDDC preconditioner for the assembled system $DA(g^{(k)})$, we have to include discrete harmonic extensions of the jump on the interface to the interior part of the subdomains. First, we define a discrete harmonic extension operator $H : \widetilde{W} \to V^h$ by

$$H(g^{(k)}) := \begin{bmatrix} 0 & -\left(DK(g^{(k)})_{II}\right)^{-1} D\widetilde{K}(g^{(k)})_{\Gamma I}^T \\ 0 & 0 \end{bmatrix}, \tag{2.19}$$

where $DK(g^{(k)})_{II}$ and $D\widetilde{K}(g^{(k)})_{\Gamma I}$ are blocks of the partially assembled tangential matrix

$$D\widetilde{K}(g^{(k)}) = \begin{bmatrix} DK(g^{(k)})_{II} & D\widetilde{K}(g^{(k)})_{\Gamma I}^T \\ D\widetilde{K}(g^{(k)})_{\Gamma I} & D\widetilde{K}(g^{(k)})_{\Gamma\Gamma} \end{bmatrix}. \tag{2.20}$$

Let us remark that the matrix $DK(g^{(k)})_{II}$ is block-diagonal, and thus applications of $\left(DK(g^{(k)})_{II}\right)^{-1}$ only require local solves on the interior parts of the subdomains and are easily parallelizable.

We also define the jump operator $P_D : \widetilde{W} \to \widetilde{W}$ by

$$P_D = I - E_D := I - \widetilde{R}\widetilde{R}_D^T. \tag{2.21}$$

This jump operator is standard in the FETI-DP theory, and it is there usually defined as $P_D = B_D^T B$; see [41, Chapter 6] and [36] for more details. Here, B is the standard jump matrix with exactly one 1 and one -1 per row, and each row corresponds to a jump between two degrees of freedom belonging to the same physical quantity but different subdomains. In B_D, the index D represents the usual scaling. We now define the second BDDC preconditioner introduced by Li and Widlund in [36, eq. (8)] as

$$M_2^{-1}(g^{(k)}) := \left(\widetilde{R}_D^T - H(g^{(k)})P_D\right)\left(D\widetilde{K}(g^{(k)})\right)^{-1}\left(\widetilde{R}_D - P_D^T H(g^{(k)})^T\right). \tag{2.22}$$

The preconditioned system $M_2^{-1}(g^{(k)})DA(g^{(k)})$ has, except for some eigenvalues equal to 0 and 1, the same eigenvalues as the standard BDDC preconditioner applied to the Schur complement; see [36, Theorem 1] for a proof. Therefore, under sufficient assumptions (see [36, Assumption 1]), the condition number κ of the preconditioned system is bounded by

$$\kappa(M_2^{-1}(g^{(k)})DA(g^{(k)})) \leq \Phi(H_s, h).$$

The exact form of the function $\Phi(H_s, h)$ depends on the problem and the choice of the primal constraints, e.g., if for a homogeneous linear elasticity problem appropriate primal constraints are chosen, then we obtain the well-known condition number bound $\Phi(H_s, h) = C(1 + \log(H_s/h))^2$. Here, H_s always denotes the maximal diameter of all subdomains and h the maximal diameter of all finite elements.

The computational kernels of the exact BDDC preconditioner M_2^{-1} consist of sparse direct solvers to compute the action a) of DK_{II}^{-1} (see (2.19)), which represents harmonic

ETNA
Kent State University and
Johann Radon Institute (RICAM)

extensions to the interior of the subdomains and b) of $D\widetilde{K}^{-1}$ (see (2.22)), which represents the solution of subdomain problems coupled in the primal unknowns. The step b) can be implemented by b_1) applying a sparse direct solver to compute the action of $D\widetilde{K}_{BB}^{-1}$ and by b_2) applying a sparse direct solver to solve the coarse problem, i.e., $D\widetilde{S}_{\Pi\Pi}^{-1}$; see, e.g., (2.28). Here, $D\widetilde{K}_{BB}$ is the non-primal block in (2.27) and $D\widetilde{S}_{\Pi\Pi}$ the Schur complement in the primal variables defined in (2.29). In the inexact BDDC variants presented in the literature, the sparse direct solvers are replaced by approximations; see the following sections.

2.5. Using inexact solvers—implemented variants. In this section, we describe those variants of nonlinear BDDC methods using inexact solvers which we implemented in our parallel software; see Table 2.2. We start with the variants proposed by Li and Widlund [36].

In this paper, we assume $D\widehat{K}(g^{(k)})$ to be a spectrally equivalent preconditioner for $D\widetilde{K}(g^{(k)})$ (as in inexact FETI-DP [32, Section 6.1]) and $D\widehat{K}(g^{(k)})_{II}$ to be spectrally equivalent to $DK(g^{(k)})_{II}$. In our numerical experiments, $D\widehat{K}(g^{(k)})^{-1}$ and $D\widehat{K}(g^{(k)})_{II}^{-1}$ are implemented by a fixed number of algebraic multigrid (AMG) V-cycles to replace $D\widetilde{K}(g^{(k)})^{-1}$ and $DK(g^{(k)})_{II}^{-1}$, respectively. Let us remark that $D\widehat{K}(g^{(k)})^{-1}$ requires an MPI parallel implementation of an AMG method, whereas for $D\widehat{K}(g^{(k)})_{II}^{-1}$ a sequential AMG implementation on each subdomain (and thus processor core or hardware thread) is sufficient, due to the favorable block-diagonal structure of $DK(g^{(k)})_{II}$.

Replacing $DK(g^{(k)})_{II}^{-1}$ in H by $D\widehat{K}(g^{(k)})_{II}^{-1}$ as suggested in [36, Section 7], we obtain the inexact discrete harmonic extension

$$\widehat{H}(g^{(k)}) := \begin{bmatrix} 0 & -\left(D\widehat{K}(g^{(k)})_{II}\right)^{-1} D\widetilde{K}(g^{(k)})_{\Gamma I}^T \\ 0 & 0 \end{bmatrix}, \tag{2.23}$$

which is an approximation of the discrete harmonic extension operator H.

We can now define a linear inexact BDDC preconditioner to be applied to the tangent system (2.17). Finally, the inexact linear lumped BDDC preconditioner is defined as

$$\widehat{M}_1^{-1}(g^{(k)}) := \widetilde{R}_D^T \left(D\widehat{K}(g^{(k)})\right)^{-1} \widetilde{R}_D \tag{2.24}$$

and the inexact linear BDDC preconditioner as

$$\widehat{M}_2^{-1}(g^{(k)}) := \left(\widetilde{R}_D^T - \widehat{H}(g^{(k)})P_D\right) \left(D\widehat{K}(g^{(k)})\right)^{-1} \left(\widetilde{R}_D - P_D^T \widehat{H}(g^{(k)})^T\right), \tag{2.25}$$

where $D\widehat{K}$ is assumed to be spectrally equivalent to $D\widetilde{K}$, and $\widehat{H}$ is the inexact harmonic extension (2.23), where we assume $D\widehat{K}(g^{(k)})_{II}$ to be spectrally equivalent to $D\widehat{K}_{II}$.

Let us remark that in NL-BDDC, the computation of $g^{(k)} := N_{NL}(\bar{u}^{(k)})$ requires the solution of (2.7) in each Newton step. This is performed by an inner Newton iteration, and in each of the Newton steps, the linear system defined in (2.10) has to be solved. A Krylov subspace method using $D\widehat{K}(g^{(k)})_{II}^{-1}$ as a preconditioner is used to solve (2.10).

Of course, also sparse direct solvers can be used for the discrete harmonic extensions, e.g., using $H(g^{(k)})$ instead of $\widehat{H}(g^{(k)})$, which defines

$$\widehat{M}_{2,EDHE}^{-1}(g^{(k)}) := \left(\widetilde{R}_D^T - H(g^{(k)})P_D\right) \left(D\widehat{K}(g^{(k)})\right)^{-1} \left(\widetilde{R}_D - P_D^T H(g^{(k)})^T\right). \tag{2.26}$$

Here, the abbreviation $EDHE$ marks the use of *exact discrete harmonic extensions*, i.e., by using a sparse direct solver. This preconditioner allows us to study the effect of replacing $D\widetilde{K}^{-1}$ by $D\widehat{K}^{-1}$ separately from the harmonic extensions.

ETNA
Kent State University and
Johann Radon Institute (RICAM)

Using a block factorization. We can represent the inverse of

$$D\widetilde{K}(g^{(k)}) = \begin{bmatrix} DK(g^{(k)})_{BB} & D\widetilde{K}(g^{(k)})_{\Pi B}^T \\ D\widetilde{K}(g^{(k)})_{\Pi B} & D\widetilde{K}(g^{(k)})_{\Pi\Pi} \end{bmatrix} \tag{2.27}$$

in block form as

$$D\widetilde{K}^{-1} = \begin{bmatrix} DK_{BB}^{-1} & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} -DK_{BB}^{-1} D\widetilde{K}_{\Pi B}^T \\ I \end{bmatrix} D\widetilde{S}_{\Pi\Pi}^{-1} \begin{bmatrix} -D\widetilde{K}_{\Pi B} DK_{BB}^{-1} & I \end{bmatrix}, \tag{2.28}$$

where $D\widetilde{S}_{\Pi\Pi}$ is the Schur complement

$$D\widetilde{S}_{\Pi\Pi} = D\widetilde{K}_{\Pi\Pi} - D\widetilde{K}_{\Pi B} \left(DK_{BB}\right)^{-1} D\widetilde{K}_{\Pi B}^T \tag{2.29}$$

representing the BDDC coarse operator. The operator $D\widetilde{S}_{\Pi\Pi}$ is identical to the coarse matrix DK_C frequently used in [16] and [1]. Defining approximations $D\widehat{S}_{\Pi\Pi}^{-1}$ for $D\widetilde{S}_{\Pi\Pi}^{-1}$ and using the block factorization (2.28), we introduce

$$\widehat{D\widetilde{K}}^{-1} = \begin{bmatrix} DK_{BB}^{-1} & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} -DK_{BB}^{-1} D\widetilde{K}_{\Pi B}^T \\ I \end{bmatrix} D\widehat{S}_{\Pi\Pi}^{-1} \begin{bmatrix} -D\widetilde{K}_{\Pi B} DK_{BB}^{-1} & I \end{bmatrix}. \tag{2.30}$$

Replacing $D\widehat{K}^{-1}$ and $\widehat{H}$ in (2.25) by $\widehat{D\widetilde{K}}^{-1}$ and H, respectively, we define the next inexact BDDC preconditioner,

$$\widehat{M}_3^{-1} := \left(\widetilde{R}_D^T - HP_D\right) \left(\widehat{D\widetilde{K}}\right)^{-1} \left(\widetilde{R}_D - P_D^T H^T\right). \tag{2.31}$$

In $\widehat{M}_3^{-1}$, we use direct sparse solvers on the subdomains, i.e., for DK_{BB}^{-1} and DK_{II}^{-1}, and thus we can afford to form the exact coarse Schur complement $D\widetilde{S}_{\Pi\Pi}$; see (2.30). The inverse $D\widetilde{S}_{\Pi\Pi}^{-1}$ is then computed inexactly using algebraic multigrid; see (2.30). This approach hence corresponds to the highly scalable irFETI-DP method [26, 33]. In this inexact BDDC, null space corrections are not necessary. Note that for null space corrections, a basis of the null space has to be known explicitly; for many applications beyond scalar diffusion this is not the case.

Note that the use of the factorization (2.28) was proposed for one of the variants of inexact FETI-DP methods; see [32, formula (8)]. The same factorization has been used earlier for the construction of inexact domain decomposition methods; see, e.g., [41, Chapter 4.3]. Let us finally list all implemented variants in Table 2.2.

2.6. Comparison with other inexact BDDC algorithms. Independently of Li and Widlund [36], three different inexact BDDC methods for linear problems have been introduced by Dohrmann in [16]. These approaches are competing with the inexact BDDC method by Li and Widlund [36] and our inexact FETI-DP methods, which were introduced in [32] at about the same time. All these efforts were driven by the scalability limitations of exact FETI-DP and BDDC methods, which became apparent on a new generation of supercomputers.

The full potential of inexact iterative substructuring approaches was then first demonstrated for the inexact FETI-DP methods, which were found to be highly scalable for linear and nonlinear problems on large supercomputers [26, 33] and which later scaled to the complete JUQUEEN and Mira supercomputers [27]. Recently, in [1], Badia et al. presented an implementation of an inexact BDDC preconditioner based on [16], which is also highly scalable.

ETNA
Kent State University and
Johann Radon Institute (RICAM)

TABLE 2.2
Implemented variants of Inexact BDDC Preconditioners for DA.

Application of $D\widetilde{K}^{-1}$	**Application of H**	**Name**
Exact computation using (2.28) and sparse direct solvers for DK_{BB}^{-1} and $\widetilde{S}_{\Pi\Pi}^{-1}$	**Lumped variant** and thus no discrete harmonic extension is used	M_1^{-1}; see (2.18)
Inexact computation using a single BoomerAMG V-cycle to approximate $D\widetilde{K}^{-1}$	**Lumped variant** and thus no discrete harmonic extension is used	$\widehat{M}_1^{-1}$; see (2.24)
Exact computation using (2.28) and sparse direct solvers for DK_{BB}^{-1} and $\widetilde{S}_{\Pi\Pi}^{-1}$	**Exact computation** using sparse direct solvers for DK_{II}^{-1}	M_2^{-1}; see (2.22)
Inexact computation using a single BoomerAMG V-cycle to approximate $D\widetilde{K}^{-1}$	**Exact computation** using sparse direct solvers for DK_{II}^{-1}	$\widehat{M}_{2,EDHE}^{-1}$; see (2.26)
Inexact computation using a single BoomerAMG V-cycle to approximate $D\widetilde{K}^{-1}$	**Inexact computation** using BoomerAMG to approximate DK_{II}^{-1}; see (2.23)	$\widehat{M}_2^{-1}$; see (2.25)
Inexact computation using (2.28) and sparse direct solvers for DK_{BB}^{-1} but a single BoomerAMG V-cycle to approximate $\widetilde{S}_{\Pi\Pi}^{-1}$; see (2.30)	**Exact computation** using sparse direct solvers for DK_{II}^{-1}	$\widehat{M}_3^{-1}$; see (2.31)

Even more recently, Zampini [44] presented an inexact BDDC code, which also implements the approach from [16] efficiently in parallel.

Note that these two recent parallel inexact BDDC implementations [1] and [44] need to apply a null space correction in every iteration of the preconditioner since they are based on [16], where a null space property [16, eq. (48)] is necessary for the two variants covered by the theory. This requires a basis for the null space to be known. In our implementation presented in this paper, however, it is not necessary to use a null space correction. Also in our FETI-DP methods with inexact subdomain solvers (iFETI-DP) [31, 32], a null space correction is not necessary.

We will review all these BDDC variants and present them in our notation in the following sections, i.e., Section 2.6.1 and Section 2.6.2.

2.6.1. Equivalence of approaches in the case of exact solvers. In this section, we will discuss that, if exact solvers are used, the approaches by Li and Widlund [36] and Dohrmann [16] are equivalent. As a consequence, for exact solvers, the approaches in Badia et al. [1] and in Zampini [44] are also equivalent since they are both based on Dohrmann [16]. Later, in Section 2.6.2, we discuss differences when inexact solvers are used.

First, we need another representation of the assembled tangential matrix, i.e.,

$$DA = \widetilde{R}^T (D\widetilde{K})\widetilde{R} = \begin{bmatrix} DK_{II} & D\widetilde{K}_{\Gamma I}^T \widetilde{R}_\Gamma \\ \widetilde{R}_\Gamma^T D\widetilde{K}_{\Gamma I} & \widetilde{R}_\Gamma^T D\widetilde{K}_{\Gamma\Gamma} \widetilde{R}_\Gamma \end{bmatrix}.$$

Keep in mind that the BDDC preconditioner for the full matrix using exact solvers in [16, eq. (5)] (and the identical one in [1, eq. (13)]) can be written in our notation as

$$M_3^{-1} = P_I + \mathcal{H}\, \widetilde{R}_D^T \left(D\widetilde{K}\right)^{-1} \widetilde{R}_D\, \mathcal{H}^T, \tag{2.32}$$

ETNA
Kent State University and
Johann Radon Institute (RICAM)

with a correction on the interior part of the subdomains $P_I : V^h \to V^h$ defined by

$$P_I := \begin{bmatrix} DK_{II}^{-1} & 0 \\ 0 & 0 \end{bmatrix}$$

and a discrete harmonic extension $\mathcal{H} : V^h \to V^h$ defined by

$$\mathcal{H} := \begin{bmatrix} 0 & -DK_{II}^{-1} D\widetilde{K}_{\Gamma I}^T \widetilde{R}_\Gamma \\ 0 & I \end{bmatrix}.$$

We will now reduce both M_3^{-1} and M_2^{-1} to the same block-system and thus show the equivalence of the two preconditioners.

The inverse of $D\widetilde{K}$ has the well-known representation

$$D\widetilde{K}^{-1} = P_I + \begin{bmatrix} -DK_{II}^{-1} D\widetilde{K}_{\Gamma I}^T \\ I \end{bmatrix} D\widetilde{S}_{\Gamma\Gamma}^{-1} \begin{bmatrix} -D\widetilde{K}_{\Gamma I} DK_{II}^{-1} & I \end{bmatrix}, \tag{2.33}$$

where the Schur complement on the interface is defined by

$$D\widetilde{S}_{\Gamma\Gamma} = D\widetilde{K}_{\Gamma\Gamma} - D\widetilde{K}_{\Gamma I} DK_{II}^{-1} D\widetilde{K}_{\Gamma I}^T.$$

Inserting (2.33) into (2.32) and by a simple computation, we obtain

$$M_3^{-1} = P_I + \begin{bmatrix} -DK_{II}^{-1} D\widetilde{K}_{\Gamma I}^T E_{D,\Gamma} \\ \widetilde{R}_{D,\Gamma}^T \end{bmatrix} D\widetilde{S}_{\Gamma\Gamma}^{-1} \begin{bmatrix} -E_{D,\Gamma}^T D\widetilde{K}_{\Gamma I} DK_{II}^{-1} & \widetilde{R}_{D,\Gamma} \end{bmatrix}, \tag{2.34}$$

using the fact that

$$\widetilde{R}_D = \begin{bmatrix} I & 0 \\ 0 & \widetilde{R}_{D,\Gamma} \end{bmatrix}$$

and defining the averaging operator $E_{D,\Gamma} = \widetilde{R}_\Gamma \widetilde{R}_{D,\Gamma}^T$.

Let us now reduce M_2^{-1} as defined in (2.22) to the same form. Combining the definition of $E_{D,\Gamma}$ and the definition of P_D from (2.21), we can write

$$P_D = \begin{bmatrix} 0 & 0 \\ 0 & I - E_{D,\Gamma} \end{bmatrix}. \tag{2.35}$$

Combining (2.35) and (2.19), we obtain by a direct computation

$$\widetilde{R}_D^T - H\, P_D = \begin{bmatrix} I & DK_{II}^{-1} D\widetilde{K}_{\Gamma I}^T (I - E_{D,\Gamma}) \\ 0 & \widetilde{R}_{D,\Gamma}^T \end{bmatrix} \tag{2.36}$$

for the first factor in (2.22). Inserting (2.33) into (2.22), we see that

$$(\widetilde{R}_D^T - H\, P_D) P_I (\widetilde{R}_D - P_D^T\, H^T) = P_I,$$

and, secondly, from (2.36) that

$$(\widetilde{R}_D^T - H\, P_D) \begin{bmatrix} -DK_{II}^{-1} D\widetilde{K}_{\Gamma I}^T \\ I \end{bmatrix} = \begin{bmatrix} -DK_{II}^{-1} D\widetilde{K}_{\Gamma I}^T E_{D,\Gamma} \\ \widetilde{R}_{D,\Gamma}^T \end{bmatrix}.$$

Combining all, we have

$$M_2^{-1} = P_I + \begin{bmatrix} -DK_{II}^{-1} D\widetilde{K}_{\Gamma I}^T E_{D,\Gamma} \\ \widetilde{R}_{D,\Gamma}^T \end{bmatrix} D\widetilde{S}_{\Gamma\Gamma}^{-1} \begin{bmatrix} -E_{D,\Gamma}^T D\widetilde{K}_{\Gamma I} DK_{II}^{-1} & \widetilde{R}_{D,\Gamma} \end{bmatrix}, \tag{2.37}$$

and thus, comparing (2.37) and (2.34), we obtain $M_2^{-1} = M_3^{-1}$.

ETNA
Kent State University and
Johann Radon Institute (RICAM)

2.6.2. Different strategies for using inexact solvers. Since we have seen that all variants [1, 16, 36] and also [44] are identical in the case of exact solvers, we can write all inexact variants in the same notation and provide a brief discussion of advantages and disadvantages in this section.

To improve readability, we will refer to the three inexact preconditioners defined by Dohrmann in [16] as $\widehat{M}_{D,1}^{-1}$ (see [16, p. 155]), $\widehat{M}_{D,2}^{-1}$, (see [16, eq. (30)]), and $\widehat{M}_{D,3}^{-1}$ (see [16, eq. (39)]), to the inexact preconditioner defined by Li and Widlund in [36] as $\widehat{M}_{LW}^{-1}$ (see [36, p. 1423]), and to the inexact variant presented by Badia, Martin, and Principe in [1] as $\widehat{M}_{BMP}^{-1}$ (see [1, eq. (13)]). Let us remark that $\widehat{M}_2^{-1} = \widehat{M}_{LW}^{-1}$.

The different variants differ in the way how the action of $D\widetilde{K}^{-1}$ and how the discrete harmonic extensions H to the interior of the subdomains are approximated; see the remarks on the computational kernels at the end of Section 2.4.

The approaches from the literature in a common notation. Let us first point out an important difference of the preconditioner $\widehat{M}_2^{-1}$ (see (2.25)) to the other inexact BDDC preconditioners. In $\widehat{M}_2^{-1}$, as also in iFETI-DP [32], the operator $D\widetilde{K}^{-1}$ is replaced by an inexact version $D\widehat{K}^{-1}$. The inexact version $D\widehat{K}^{-1}$ can be constructed from the standard block factorization

$$D\widetilde{K}^{-1} = \begin{bmatrix} DK_{BB}^{-1} & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} -DK_{BB}^{-1} D\widetilde{K}_{\Pi B}^T \\ I \end{bmatrix} D\widetilde{S}_{\Pi\Pi}^{-1} \begin{bmatrix} -D\widetilde{K}_{\Pi B} DK_{BB}^{-1} & I \end{bmatrix}, \tag{2.38}$$

where $D\widetilde{S}_{\Pi\Pi} = D\widetilde{K}_{\Pi\Pi} - D\widetilde{K}_{\Pi B} DK_{BB}^{-1} D\widetilde{K}_{\Pi B}^T$ is the coarse problem. But instead of using (2.38), a preconditioner for $D\widetilde{K}$ can also directly be constructed; see the discussion in [32, Section 4] and the numerical results for BoomerAMG applied directly to $D\widetilde{K}$ in [31, Table 5] for iFETI-DP; also see [36, p. 1423].

In contrast, in all other methods, i.e., $\widehat{M}_3^{-1}$, $\widehat{M}_{BMP}^{-1}$, and $\widehat{M}_{D,i}^{-1}$, $i = 1, 2, 3$, the operator is first split into a fine and a coarse correction using (2.38).

By choosing an approximation $D\overline{K}_{BB}$ of DK_{BB}, an approximation to the primal Schur complement $D\widetilde{S}_{\Pi\Pi}$ can be defined, i.e.,

$$D\overline{S}_{\Pi\Pi} = D\widetilde{K}_{\Pi\Pi} - D\widetilde{K}_{\Pi B} \left(D\overline{K}_{BB}\right)^{-1} D\widetilde{K}_{\Pi B}^T.$$

This approximate coarse matrix $D\overline{S}_{\Pi\Pi}$ is denoted by $D\widetilde{K}_C$ in [16, eq. (20)]. Defining an approximation $D\widehat{\overline{S}}_{\Pi\Pi}^{-1}$ for $D\overline{S}_{\Pi\Pi}^{-1}$ and using (2.38), we obtain

$$D\overline{K}^{-1} = \begin{bmatrix} D\overline{K}_{BB}^{-1} & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} -D\overline{K}_{BB}^{-1} D\widetilde{K}_{\Pi B}^T \\ I \end{bmatrix} D\widehat{\overline{S}}_{\Pi\Pi}^{-1} \begin{bmatrix} -D\widetilde{K}_{\Pi B} D\overline{K}_{BB}^{-1} & I \end{bmatrix},$$

which can be viewed as an approximation of $D\widetilde{K}^{-1}$.

The first inexact preconditioner introduced by Dohrmann [16, p. 155] then is

$$\widehat{M}_{D,1}^{-1} := \left(\widetilde{R}_D^T - HP_D\right) \left(D\overline{K}\right)^{-1} \left(\widetilde{R}_D - P_D^T H^T\right),$$

the second [16, eq. (30)] is

$$\widehat{M}_{D,2}^{-1} := \left(\widetilde{R}_D^T - \widehat{H}P_D\right) \left(D\overline{K}\right)^{-1} \left(\widetilde{R}_D - P_D^T \widehat{H}^T\right),$$

where, additionally, an approximate discrete harmonic extension $\widehat{H}$ is used as in (2.23).

ETNA
Kent State University and
Johann Radon Institute (RICAM)

Similarly, we can write [1, eq. (13)]) as

$$\widehat{M}^{-1}_{BMP} := \left(\widetilde{R}_D^T - \widehat{H} P_D\right) \left(D\overline{\overline{K}}\right)^{-1} \left(\widetilde{R}_D - P_D^T \widehat{H}^T\right),$$

where

$$D\overline{\overline{K}}^{-1} = \begin{bmatrix} D\overline{K}^{-1}_{BB} & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} -D\overline{K}^{-1}_{BB} D\widetilde{K}^T_{\Pi B} \\ I \end{bmatrix} D\widehat{\overline{S}}^{-1}_{\Pi\Pi} \begin{bmatrix} -D\widetilde{K}_{\Pi B} D\overline{K}^{-1}_{BB} & I \end{bmatrix}.$$

Here, $D\widehat{\overline{S}}^{-1}_{\Pi\Pi}$ is an approximation to $D\overline{\overline{S}}^{-1}_{\Pi\Pi}$, while

$$D\overline{\overline{S}}_{\Pi\Pi} := D\overline{S}_{\Pi\Pi} + D\widetilde{K}_{\Pi B} D\overline{K}^{-1}_{BB} \left(DK_{BB} D\overline{K}^{-1}_{BB} - I\right) D\widetilde{K}^T_{\Pi B}$$

itself is an approximation to $D\overline{S}_{\Pi\Pi}$ or, respectively, to $D\widetilde{S}_{\Pi\Pi}$. Let us remark that in [1] only sequential AMG implementations are used, while we apply a parallel AMG on a subset of cores to construct $D\widehat{S}^{-1}_{\Pi\Pi}$; see also Section 3.3.

Note that, for reliable convergence, in general these three inexact BDDC preconditioners need to use an additional null space correction, which we have omitted in the presentation here; see [16, Section 4.2].

Finally, the third inexact BDDC preconditioner by Dohrmann [16, eq. (39)] is

$$\widehat{M}^{-1}_{D,3} := \left(\widetilde{R}_D^T - \overline{H} P_D\right) \left(D\overline{K}\right)^{-1} \left(\widetilde{R}_D - P_D^T \overline{H}^T\right).$$

This variant is not covered by the theory in [16] but is reported to perform well without a null space correction. Here, $\overline{H}$ is an approximation to the discrete harmonic extension defined by

$$\overline{H} := \begin{bmatrix} I - \left(D\widehat{K}_{II}\right)^{-1} DK_{II} & -\left(D\widehat{K}_{II}\right)^{-1} D\widetilde{K}^T_{\Gamma I} \\ 0 & 0 \end{bmatrix}.$$

A brief discussion of the different approaches. Let us briefly discuss the different inexact solvers needed in the inexact BDDC methods. The matrix DK_{BB} is block-diagonal, and thus a few V-cycles of a sequential AMG method applied to each block separately is sufficient to construct $D\overline{K}^{-1}_{BB}$. The same is true for the approximate harmonic extensions $\widehat{H}$ and $\overline{H}$, which can be computed by applying AMG to each block of DK_{II}. If a coarse problem is explicitly formed, then, for scalability, a parallel AMG method should be applied to define $D\widehat{\overline{S}}^{-1}_{\Pi\Pi}$ or $D\widehat{S}^{-1}_{\Pi\Pi}$ as, e.g., in [26, 33]. As common in parallel multigrid methods, this parallel AMG method can run on a small subset of computing cores. If the AMG methods is directly applied to $D\widetilde{K}$ as in $\widehat{M}_2^{-1}$, then a highly scalable parallel AMG method operating on all cores is mandatory.

In the inexact BDDC variants making use of the block factorization (2.38), the coarse problem can be solved in parallel to the local subdomain problems.

To prove a condition number bound for the inexactly preconditioned systems with preconditioners $\widehat{M}^{-1}_{D,i}$, $i = 1, 2$, $\widehat{M}^{-1}_{BMP}$, it is necessary that the matrices

$$\begin{bmatrix} D\overline{K}_{BB} & DK^T_{\Pi B} \\ DK_{\Pi B} & DK_{\Pi\Pi} \end{bmatrix}$$

and DK are spectrally equivalent and thus need to have the same null space. To guarantee this, an explicit null space correction (kernel correction) in $D\overline{K}_{BB}$ is necessary, which can be performed locally on each subdomain; see [16, Section 4.2]. This is a disadvantage of the approaches introduced in [16] since the null space of DK has to be explicitly known.

ETNA
Kent State University and
Johann Radon Institute (RICAM)

3. Implementation remarks. We have implemented the inexact nonlinear BDDC methods using PETSc 3.6.4 [5] and MPI. For all inexact solves, we use one or two V-cycles of BoomerAMG [21]; see Section 3.3 for details. In case of exact solves, we always apply the sparse direct solver UMFPACK [12]. We distribute all subdomains Ω_i, $i = 1, \dots, N$, to all available MPI ranks. For simplicity, we assume throughout the following discussion that we have exactly one subdomain Ω_i per process rank$_i$, even though our software is capable of handling several subdomains on each MPI rank. We always use Krylov subspace methods such as CG (Conjugate Gradient) or GMRES (Generalized Minimal Residual) to solve all linear systems. Therefore, it is sufficient to describe the application of the BDDC operators to appropriate vectors. Let us also remark that our implementation uses building blocks from our highly scalable implementation of the inexact nonlinear FETI-DP method presented in [28] which itself is based on [24].

3.1. Discussion of the parallel data structures. In every Krylov iteration the linear preconditioner is applied to an MPI-parallel and fully assembled residual vector $\bar{b} \in V^h$. In our software package, $\bar{b}$ is distributed to all available MPI ranks such that communication is minimized. More precisely, each entry of $\bar{b}$, which belongs to a degree of freedom in the interior part of a subdomain Ω_k, is stored on the corresponding rank$_k$. Entries of $\bar{b}$ belonging to the interface between several subdomains are stored on a rank handling one of the neighboring subdomains. Similarly, we distribute primally assembled vectors $\tilde{b} \in \widetilde{W}$ to all MPI ranks. Here, all entries corresponding to uncoupled, non-primal degrees of freedom on the subdomain Ω_k are again stored on rank$_k$. The primal variables are distributed evenly to all or a subset of all MPI ranks.

The application of the operators $\overline{R} : \widetilde{W} \to W$ and $R : V^h \to W$ on $\tilde{b}$ and $\bar{b}$, respectively, are implemented using PETSc's *VecScatter* class. Therefore, applications of $\overline{R}$ and R cause an *MPI_Scatter* operation, while applications of $\overline{R}^T$ and R^T cause an *MPI_Gather* operation on all MPI ranks. In general, R is more expensive since it causes communication in all interface variables, while $\overline{R}$ only causes communication in the few primal variables. The restriction $\widetilde{R} : V^h \to \widetilde{W}$ is not implemented explicitly. Instead, we always connect R and $\overline{R}^T$ in series and apply the necessary scaling in the primal degrees of freedom. In general, all scalings marked with an index D are performed using the PETSc function *VecPointwiseMult*. This is done completely locally and in parallel on each subdomain, i.e., in the space W. The jump operator $P_D = I - \widetilde{R}\widetilde{R}_D^T$ can be implemented using those building blocks.

In $\widehat{M}_2^{-1}$ and $\widehat{M}_{2,EDHE}^{-1}$, the matrix $D\widetilde{K}(g^{(k)})$ has to be assembled explicitly as an MPI parallel matrix of PETSc's type *MPIAIJ* using the subdomain tangential matrices $DK_i(R_i g^{(k)})$, $i = 1, \dots, N$. The matrix $D\widetilde{K}(g^{(k)})$ is distributed to all MPI ranks in a row-wise fashion, equivalently to the distribution of $\tilde{b}$; see also [28] for more details. Therefore, this assembly process requires only communication in the primal variables. For larger numbers of MPI ranks, this is currently more efficient in PETSc than the complete assembly of $DA(g^{(k)})$, which is not necessary in our software and always avoided. To avoid unnecessary buffer space during the assembly of $D\widetilde{K}(g^{(k)})$, we precompute an estimate of the number of nonzero entries in each row of $D\widetilde{K}(g^{(k)})$ using only local information obtained from an analysis of the sparsity patterns of $DK_i(R_i g^{(k)})$, $i = 1, \dots, N$. This way, a nearly optimal preallocation for the *MPIAIJ* matrix can be used. It is also possible to overlap the communication of the assembly process with local work in the BDDC setup process. For the inexact solve $D\widehat{K}(g^{(k)})^{-1}$, we use one or two V-cycles of MPI parallel BoomerAMG [21] applied to $D\widetilde{K}(g^{(k)})$. For the inexact harmonic extension $\widehat{H}$, we use a sequential BoomerAMG on each subdomain, applied to the interior part of the local tangential matrix. For details about the AMG setup, see Section 3.3.

ETNA
Kent State University and
Johann Radon Institute (RICAM)

When using $\widehat{M_3}^{-1}$, we use the same building blocks as described above since solely the application of BoomerAMG to $D\widetilde{K}(g^{(k)})$ is replaced by (2.30). Therefore, instead of assembling $D\widetilde{K}(g^{(k)})$ as described above, we now assemble the smaller Schur complement matrix $D\widetilde{S}_{\Pi\Pi}(g^{(k)})$ on a smaller subset of MPI ranks, e.g., one percent of the available MPI ranks. Analogously, a restriction operator $\overline{R} : \widetilde{W} \to W$ is no longer needed, and instead we have implemented a primal restriction operator $R_\Pi : \widetilde{W}_\Pi \to W_\Pi$ using again PETSc's *VecScatter* class. Here, W_Π is the space of the unassembled local primal variables on the subdomains. Finally, as before, for the inexact solve $D\widehat{S}^{-1}_{\Pi\Pi}$, we use parallel BoomerAMG. All remaining operations in (2.30) are completely local to each MPI rank.

For the exact preconditioner M_2^{-1}, we decided to use the same building blocks as for $\widehat{M_3}^{-1}$, but we create a local copy of $D\widetilde{S}_{\Pi\Pi}(g^{(k)})$ on each MPI rank and apply a direct solver for an exact solution of the coarse problem.

3.2. Application of the tangential matrix. An application of the operator

$$DA(g^{(k)}) = R^T DK(Rg^{(k)})R$$

to a vector $\bar{b} \in V^h$ can be performed in two different fashions. The first one is straightforward, namely scattering $\bar{b}$ to the local subdomains by using R, then applying $DK_i(R_i g^{(k)})$, $i = 1, \ldots, N$, on each subdomain in parallel, and finally gathering and assembling the local results using R^T. This variant is efficient and highly scalable, and we always choose it when using $\widehat{M_3}^{-1}$. Considering $\widehat{M_2}^{-1}$, we need to store the local matrices $DK_i(R_i g^{(k)})$, $i = 1, \ldots, N$, in addition to $D\widetilde{K}(g^{(k)})$. To increase the memory efficiency, we propose a second variant utilizing the representation $DA(g^{(k)}) = \widetilde{R}^T D\widetilde{K}(g^{(k)})\widetilde{R}$. Here, we can delete all local copies of $DK_i(R_i g^{(k)})$, $i = 1, \ldots, N$, after $D\widetilde{K}(g^{(k)})$ has been built. Let us remark that the amount of communication as well as the local work is slightly higher compared to the first variant. We implemented both approaches, and, in the case of $\widehat{M_2}^{-1}$, the user can choose if he or she can afford the additional memory requirements of the potentially faster variant.

3.3. Inexact solves with BoomerAMG. Let us finally discuss the BoomerAMG setup used. Since we are concerned with systems of PDEs, e.g., elasticity or plasticity problems, we always use nodal coarsening approaches throughout this paper for $D\widehat{K}(g^{(k)})^{-1}$, $D\widehat{S}_{\Pi\Pi}(g^{(k)})^{-1}$, or $D\widehat{K}(g^{(k)})^{-1}_{II}$. We always use the highly scalable HMIS coarsening [14] in combination with the extended+i interpolation [13, 43]. Additionally, the interpolation matrix is truncated to a maximum of three entries per row. This guarantees an appropriate operator complexity. A threshold of 0.3 is used for the detection of strong couplings. We can also use the GM (global matrix) or the LN (local neighborhood) interpolation approach (see [4]), which allow to interpolate near null space vectors exactly. This approach can help to improve scalability and performance of AMG for elasticity problems significantly [3]. For elasticity and plasticity problems, the rigid body motions are computed, transformed if a change of basis is used in the BDDC coarse space, and restricted to $\widetilde{W}$. This leads to a slightly more expensive setup but improves numerical scalability. For the subdomain Dirichlet problems, the GM or LN approaches are not used since, for problems with large Dirichlet boundary, the standard approaches are sufficient and have lower setup cost. For an overview of the performance of the different AMG approaches for linear elasticity problems; see also [3]. A detailed discussion on the effect of the GM approach used in $\widehat{M_2}^{-1}$ and $\widehat{M}^{-1}_{2,EDHE}$ is presented in [29].

4. Numerical results.

4.1. Model problems and geometries. Since our focus is on structural mechanics, in this paper we consider three different mechanical problems. First, we use the classical *linear*

ETNA
Kent State University and
Johann Radon Institute (RICAM)

elastic material model in order to demonstrate weak scaling of our implementation for linear problems. Let us remark that in the linear case both NL-BDDC and NK-BDDC are the same method. To compare the different nonlinear approaches, we consider a hyperelastic *Neo Hooke* model and, finally, a *J2 elasto-plasticity* problem.

4.2. Computational platforms. We perform our computations on a Tier-3 and a Tier-1/Tier-0 supercomputer of the German High Performance Computing Pyramid and on Theta:

- Theta: 231 424 cores (Cray XC40, Intel Xeon Phi 7230 64C 1.3 GHz); 5.8 PFlops; operated by Argonne National Laboratories, USA; ranked 16th in the current TOP500 list (June, 2017).
- JUQUEEN (Tier-1/0): 458 752 Blue Gene/Q cores (PowerPC A2 1.6 GHz; 16 cores and 16 GB per node); 5.0 PFlops; operated by Jülich Supercomputing Center (JSC) providing computing time for Germany and Europe; ranked 21th in the current TOP500 list (June, 2017).
- MagnitUDE (Tier-3): 13 536 cores (Broadwell XEON E5-2650v4 12C 2.2GHz; 24 cores and 72 GB per node); 476.5 TFlops NEC Cluster; operated by Center for Computational Sciences and Simulation (CCSS) of the Universität Duisburg-Essen (UDE) providing computing ressources for UDE; TOP500 rank 430 (June, 2017).

4.3. Weak scalability for linear problems. We first apply our inexact preconditioners M_2^{-1}, $\widehat{M}_2^{-1}$, $\widehat{M}_{2,EDHE}^{-1}$, and $\widehat{M}_3^{-1}$ to linear problems in two and three dimensions. This avoids effects from a varying number of Newton iterations. We do not present results for $\widehat{M}_1^{-1}$ since, generally, we do not recommend its use except when being very scarce on memory.

4.3.1. Linear elastic beam bending problem on JUQUEEN. Let us first take a detailed look at the preconditioner $\widehat{M}_2^{-1}$. In Table 4.1 and the corresponding Figure 4.1, we present weak scalability for a problem of a two-dimensional beam scaled up from 32 to 524 288 MPI ranks and subdomains using 16 to 262 144 computational cores of the JUQUEEN BlueGene/Q supercomputer, respectively. We thus use two MPI ranks per computational core. This setup makes the most efficient use of the hardware threads for our kinds of problems [25, 27].

The linear beam domain $\Omega = [0, 8] \times [0, 1]$ is fixed on the left and a volume force is applied in y-direction. Using a relative stopping tolerance of 1e-8 in the CG method, between 19 and 23 CG iterations are necessary until convergence, which confirms the numerical scalability of the method. Our parallel implementation is highly scalable and shows a parallel efficiency of 80% using 524 288 MPI ranks to solve a problem with more than 26 billion degrees of freedom. In Figure 4.2, we present detailed timings for the setup of $\widehat{M}_2^{-1}$ (Figure 4.2, left) and for a single preconditioned CG iteration (Figure 4.2, right). We here use the GM interpolation in BoomerAMG; see Section 3.3. The scalability of the setup is reasonable with an expected increase in the runtime of the parallel BoomerAMG setup for the construction of $D\widehat{K}(g^{(k)})^{-1}$ as well as a slight increase in the runtime of the communication intense parts such as the construction of the *VecScatters*. As it can be seen in Figure 4.2 (right), the scalability of the average runtime of a single preconditioned CG iteration is almost perfect, and the larger part of time is spent in the different AMG V-cycles.

4.3.2. Linear elastic cube on JUQUEEN. As a three-dimensional test problem, we consider a linear elastic cube $\Omega = [0, 1] \times [0, 1] \times [0, 1]$ with Dirichlet boundary conditions on $\partial\Omega$. A remarkable 74% of parallel efficiency is achieved for $\widehat{M}_2^{-1}$; see Table 4.2 and the corresponding Figure 4.6. Weak scaling tests from 64 MPI ranks and 1.5 million degrees of freedom to 262 144 MPI ranks and 6.3 billion degrees of freedom are presented in Table 4.2 and Figure 4.6. In Figure 4.7 (left), as in two dimensions, an increase in the runtime of the

ETNA
Kent State University and
Johann Radon Institute (RICAM)

communication intense parts of the BDDC setup (parallel AMG setup, *VecScatter* setup) can be observed, while the scalability of the average runtime of a single CG iteration is again nearly perfect; see Figure 4.7 (right).

4.3.3. Linear elastic beam bending problem on Theta. Our problem setup is similar to the one used in Table 4.1 but using a slightly longer beam $\Omega = [0,16] \times [0,1]$ to better fit the number of available cores. Also the number of degrees of freedom handled per core is much larger compared to the computations on JUQUEEN—the largest problem on 193K Knights Landing cores consists of 30 billion degrees of freedom. On Theta, we use one MPI process for each core.

We provide a brief comparison of the different preconditioners M_2^{-1}, $\widehat{M}_2^{-1}$, $\widehat{M}_{2,EDHE}^{-1}$, and $\widehat{M}_3^{-1}$ on the Knights Landing supercomputer Theta in Figure 4.3. Theta is ALCF's (Argonne Leadership Computing Facility) stepping stone to the next generation 200 petaflop Aurora supercomputer. For comparison, we also added the exact BDDC preconditioner M_2^{-1}, where a sequential copy of the coarse problem is factorized exactly on each core. This version runs out of memory beyond 65K cores.

We conclude from Figure 4.3 that the performance of all inexact preconditioners, except for $\widehat{M}_3^{-1}$ without GM interpolation, is very similar (both in terms of GMRES iterations as well as computing time) to the exact BDDC preconditioner up to 65K cores, but the inexact methods continue to scale when the exact methods fail. All three inexact variants $\widehat{M}_2^{-1}$, $\widehat{M}_{2,EDHE}^{-1}$, and $\widehat{M}_3^{-1}$ equipped with BoomerAMG with the GM approach show a very similar performance with respect to time range and number of GMRES iterations. The preconditioner $\widehat{M}_3^{-1}$ without GM interpolation, however, suffers from the use of the standard interpolation designed for systems of PDEs whose null spaces exclusively consist of the translations. This interpolation is inappropriate for the bending problem considered here, resulting in bad performance.

We also include results for heterogeneous compressible ($\nu = 0.3$) linear elasticity problems. Therefore, we consider stiff inclusions ($E = 210\,000$) in a soft matrix material ($E = 210$). Each of the circular inclusions is embedded in a single subdomain. Here, $\widehat{M}_3^{-1}$ outperforms $\widehat{M}_2^{-1}$ and $\widehat{M}_{2,EDHE}^{-1}$ in terms of GMRES iterations and runtime since the sparse direct solvers can optimally resolve the heterogeneity on the subdomains and the parallel Boomer AMG is only used on the coarse problem in $\widehat{M}_3^{-1}$; see Figure 4.4 for the results.

4.3.4. Comparison of the performance on Intel KNL and IBM BG/Q. We also include a comparison of the performance for Intel Xeon Phi Knights Landing (Theta) and IBM BlueGene/Q (JUQUEEN); see Figure 4.5. Here, KNL is approximately 1.4 times faster for M_2^{-1} as well as $\widehat{M}_3^{-1}$ and more than 2 times faster for $\widehat{M}_2^{-1}$ since the performance of BoomerAMG is much better on KNL compared to the BlueGene/Q. Considering the sparse direct solver UMFPACK, the performance gap is smaller. The weak scaling behavior is similar on both machines and only slightly better on the KNL. Let us remark that for these tests we use a single MPI rank per core on both machines.

4.3.5. Linear elastic brick on Theta. We also include results for a domain $\Omega = [0,2] \times [0,1] \times [0,1]$ obtained on Theta, again comparing $\widehat{M}_2^{-1}$ and $\widehat{M}_3^{-1}$ for the same linear homogeneous elasticity problem; see Figure 4.8. The weak scaling test starts on 128 Knights Landing cores with a problem size of 3.1 million degrees of freedom and is scaled up to 128 000 Knights Landing cores with a problem size of 3.1 billion degrees of freedom. Especially $\widehat{M}_3^{-1}$ is numerically stable, and the obtained parallel efficiency of 68% is acceptable. A better scalability is prevented by the AMG setup, which is not scaling optimally on Knights Landing for a three-dimensional elasticity problem, at least with our choice of interpolation

ETNA
Kent State University and
Johann Radon Institute (RICAM)

TABLE 4.1
Linear 2D beam bending $\Omega = [0,8] \times [0,1]$*; using* $\widehat{M}_2^{-1}$*; homogeneous and compressible linear elastic material with* $E = 210$ *and* $\nu = 0.3$*; P2 finite elements and* $H_s/h = 80$*; one V-cycle of BoomerAMG with nodal HMIS coarsening and extended+i interpolation is used in all AMG applications, and additionally the GM approach is used in all parallel AMG applications; the BDDC coarse space consists of vertex constraints;* **It.** *denotes the number of CG iterations; the baseline of the parallel efficiency* **Eff.** *is the fastest time to solution on* 32 *MPI ranks (1 node); using two MPI ranks for each core.*

# MPI ranks	D.o.f.	It.	Time to Solution	Eff.	Time Assembly $K^{(i)}$	Time Setup $\widehat{M}_2^{-1}$	Time CG
32	1 644 162	20	48.6s	100%	4.5s	9.9s	34.2s
128	6 565 122	21	50.8s	96%	4.5s	10.2s	36.1s
512	26 237 442	21	50.7s	96%	4.5s	10.1s	36.1s
2 048	104 903 682	19	47.5s	102%	4.5s	10.2s	32.8s
8 192	419 522 562	20	49.4s	98%	4.5s	10.4s	34.5s
32 768	1 677 905 922	19	48.2s	101%	4.5s	10.7s	33.0s
131 072	6 711 255 042	21	53.1s	92%	4.5s	11.7s	36.8s
524 288	26 844 282 882	23	60.6s	80%	4.5s	15.3s	40.7s

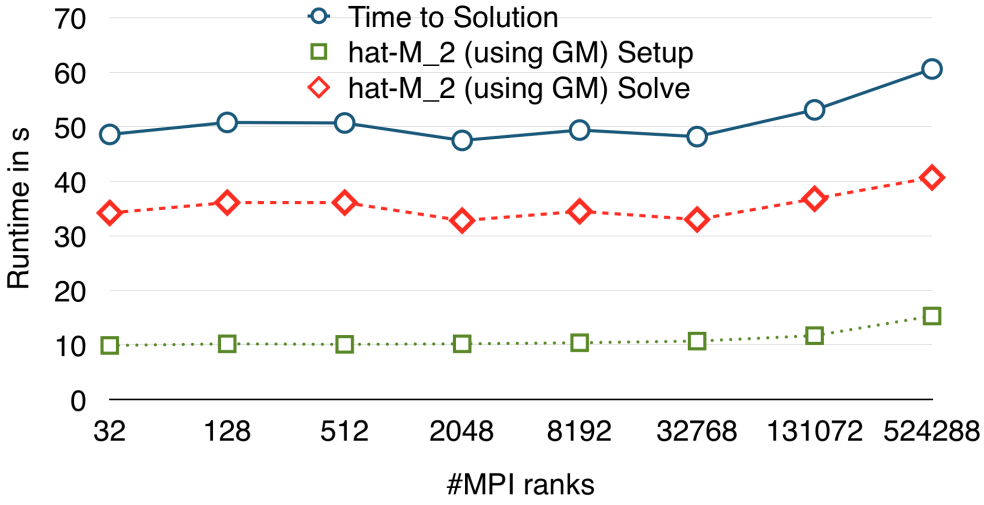

FIG. 4.1. *Weak scaling of* $\widehat{M}_2^{-1}$ *for a linear elasticity problem in two dimensions. See Table 4.1 for corresponding results and problem details.*

(ext+i) and coarsening (HMIS) strategies. This effect is even more significant for $\widehat{M}_2^{-1}$, where BoomerAMG is applied to the complete matrix $D\widetilde{K}$; see Figure 4.8 (right). The latter one is the potentially faster preconditioner since the sparse direct solver UMFPACK is comparably slow for three-dimensional elasticity problems. Therefore, the preconditioner $\widehat{M}_2^{-1}$, which does not use any sparse direct solvers, is 2.3 times faster than $\widehat{M}_3^{-1}$ on 128 cores.

4.4. Nonlinear hyperelasticity with increasing load. After the sanity check of our BDDC preconditioners M_2^{-1}, $\widehat{M}_2^{-1}$, $\widehat{M}_{2,EDHE}^{-1}$, and $\widehat{M}_3^{-1}$ for linear problems, we consider the nonlinear preconditioner N_{NL} (see (2.8)) applied to nonlinear problems. We will observe

ETNA
Kent State University and
Johann Radon Institute (RICAM)

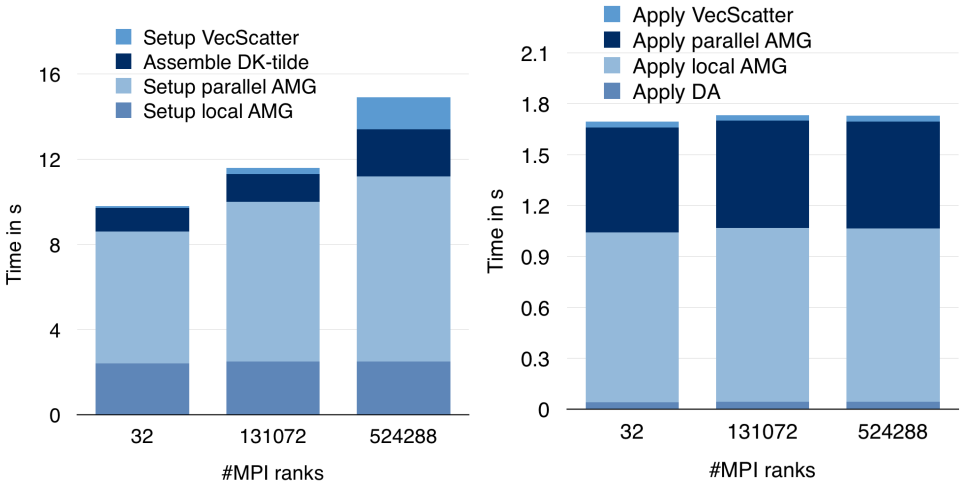

FIG. 4.2. *Weak scaling of $\widehat{M}_2^{-1}$ for a linear elasticity problem in two dimensions.* **Left:** *setup of the linear BDDC preconditioner split into the setup of the local AMGs $D\widehat{K}_{II}^{-1}$, the setup of the parallel AMG $D\widehat{K}^{-1}$, the assembly of $D\widetilde{K}$, and the setup of the different VecScatter.* **Right:** *average time per CG iteration split into the application of DA, the application of $D\widehat{K}_{II}^{-1}$, the application of $D\widehat{K}^{-1}$, and the application of the different VecScatter operations.*

TABLE 4.2
Linear Elastic 3D Cube $\Omega = [0,1] \times [0,1] \times [0,1]$*; using $\widehat{M}_2^{-1}$; P1 finite elements and $H_s/h = 20$; homogeneous and compressible linear elastic material with $E = 210$ and $\nu = 0.3$; one V-cycle of BoomerAMG with nodal HMIS coarsening and extended+i interpolation is used for all AMG applications; the BDDC coarse space consists of vertex constraints and continuity in the midpoint of each edge;* **It.** *denotes the number of CG iterations; the baseline of the parallel efficiency* **Eff.** *is the fastest time to solution on 64 MPI ranks (2 nodes); using two MPI ranks for each core; discretized with piecewise linear finite elements (P1).*

# MPI ranks	D.o.f.	It.	Time to Solution	Eff.	Time Assembly $K^{(i)}$	Time Setup $\widehat{M}_2^{-1}$	Time CG
64	1 594 323	26	63.2s	100%	22.9s	10.3s	30.0s
512	12 519 843	29	67.2s	94%	22.9s	10.3s	33.9s
4 096	99 228 483	31	71.3s	89%	23.0s	11.7s	36.6s
32 768	790 124 163	32	75.0s	84%	23.1s	13.5s	38.4s
262 144	6 306 213 123	35	85.4s	74%	23.1s	19.4s	43.0s

an enlarged radius of convergence and increased efficiency compared to the corresponding standard Newton-Krylov BDDC iterations.

In this section, we decided to use exclusively the linear preconditioner $\widehat{M}_2^{-1}$, which makes the most efficient use of the available memory. Throughout this and the following sections, we always use a relative stopping tolerance of 1e-6 for the CG method in all linear solves. We stop the inner Newton iteration on a subdomain in NL-BDDC if the 2-norm of the restriction of $\delta g_{I,l}^{(k)}$ to the subdomain is smaller than 1e-6; see (2.10) for the definition of $\delta g_{I,l}^{(k)}$. The outer Newton iteration is stopped if the 2-norm of the Newton update is smaller than 1e-8 in three dimensions or, respectively, 1e-6 in two dimensions. In general, the choice of the stopping criterion for the inner Newton iteration can have a significant impact on the overall performance of NL-BDDC. Here, solving to a stopping tolerance of 1e-6 is always

ETNA
Kent State University and
Johann Radon Institute (RICAM)

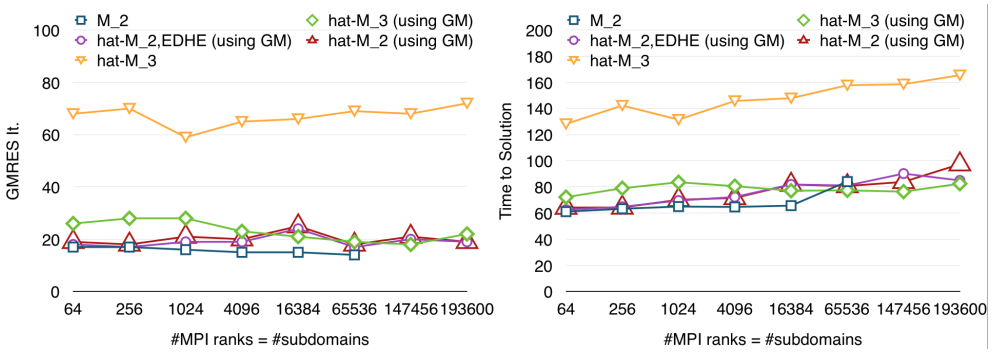

FIG. 4.3. **Linear 2D beam bending** $\Omega = [0, 16] \times [0, 1]$ *computed on Theta; homogeneous and compressible linear elastic material with* $E = 210$ *and* $\nu = 0.3$*; P2 finite elements and* $H_s/h = 140$*; one V-cycle of BoomerAMG with nodal HMIS coarsening and extended+i interpolation is used in all AMG applications, and additionally the GM approach is used in most parallel AMG applications (variants marked with GM in the legend); the BDDC coarse space consists of vertex constraints; using one MPI rank per core; comparison of* M_2^{-1}, $\widehat{M}_2^{-1}$, $\widehat{M}_{2,EDHE}^{-1}$, *and* $\widehat{M}_3^{-1}$*; largest problem has 30 billion degrees of freedom.* **Left:** *number of GMRES iterations.* **Right:** *total time to solution.*

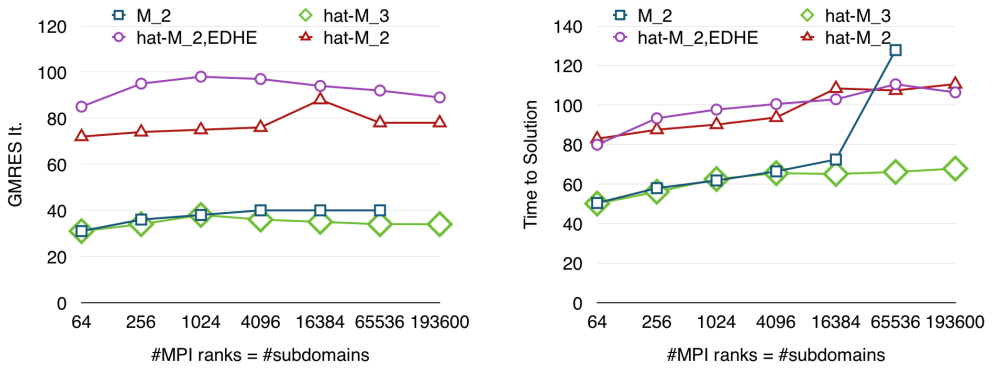

FIG. 4.4. **Linear Heterogeneous Elasticity 2D problem** $\Omega = [0, 1] \times [0, 1]$ *computed on Theta; heterogeneous and compressible linear elastic material with* $E = 210\,000$ *and* $\nu = 0.3$ *in circular inclusions and* $E = 210$ *and* $\nu = 0.3$ *in matrix material; P2 finite elements and* $H_s/h = 120$*; one V-cycle of BoomerAMG with nodal HMIS coarsening and extended+i interpolation is used in all AMG applications; the BDDC coarse space consists of vertex constraints; using one MPI rank per core; discretized with piecewise quadratic finite elements (P2); comparison of* M_2^{-1}, $\widehat{M}_2^{-1}$, $\widehat{M}_{2,EDHE}^{-1}$, *and* $\widehat{M}_3^{-1}$*; largest problem has 22.3 billion degrees of freedom.* **Left:** *number of GMRES iterations.* **Right:** *total time to solution.*

accurate enough. Even faster times to solution and increased robustness can be achieved by using an advanced stopping strategy already introduced in [30, Section 4]. Let us remark that we always use a step length of $\alpha^{(k)} = 1$ in our experiments. We can show that NL-BDDC allows for larger load steps compared to NK-BDDC in the case of heterogeneous nonlinear hyperelasticity problems. We consider a Neo-Hooke material model with stiff inclusions ($E = 210\,000$) in a soft matrix material ($E = 210$). Each inclusion is embedded in a single subdomain.

In Table 4.3, we present the results of a deformation of a square built of a heterogeneous Neo-Hooke material. We test different loads from 0.01 up to 0.8 and apply the deformation

$$F = \begin{bmatrix} \text{load} & 0 \\ 0 & -\text{load} \cdot \nu \end{bmatrix}$$

ETNA
Kent State University and
Johann Radon Institute (RICAM)

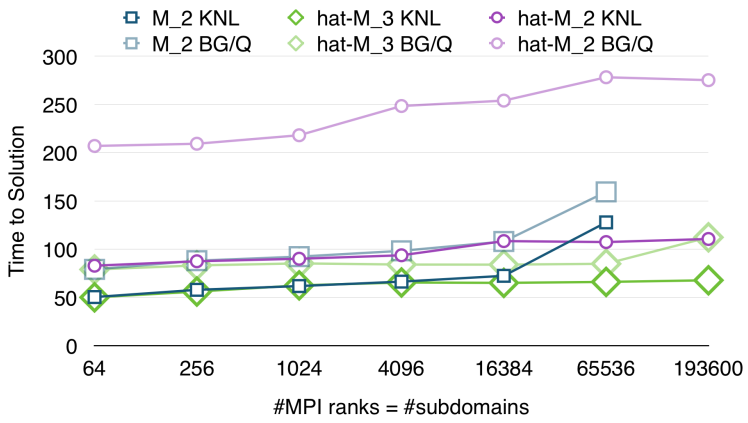

FIG. 4.5. **Comparison KNL and BG/Q**; *see Figure 4.4 for a problem description.*

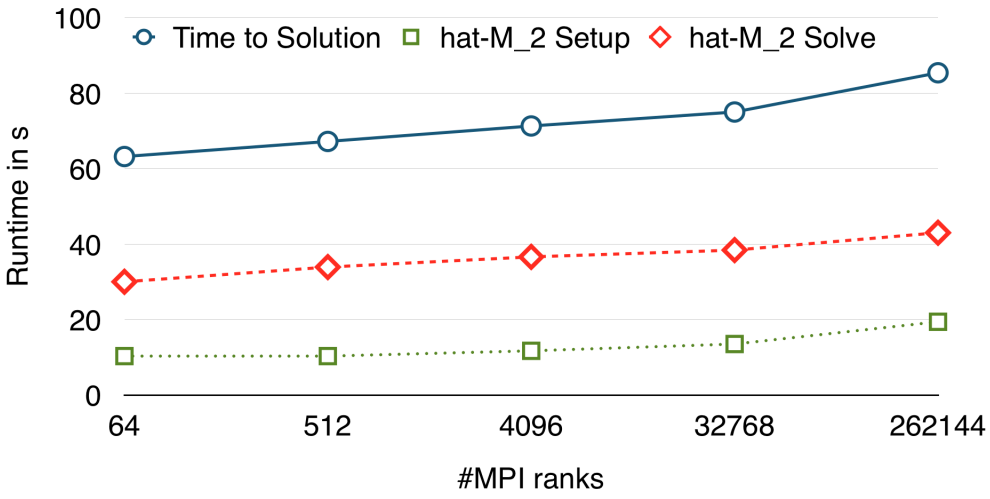

FIG. 4.6. *Weak scaling of* $\widehat{M_2}^{-1}$ *for a linear elasticity problem in three dimensions. See Table 4.2 for corresponding results and details.*

on the boundary, always in a single load step. From Table 4.3, we learn that NL-BDDC is always the faster method for both problem sizes (64 and 1 024 subdomains) since it reduces the number of global Newton steps and GMRES iterations. Moreover, NK-BDDC fails for loads larger than 0.5, while NL-BDDC remains stable. This effect illustrates the enlarged convergence radius of NL-BDDC. A visual representation of the results for 1 024 subdomains and MPI ranks is presented in Figure 4.10 and Figure 4.11. Let us remark that the local nonlinear solves (the inner iterations) stay robust even for larger loads. The local problems are easier to solve than the global problem due to the smaller problem size, the smaller number of strong nonlinearities, and, since always a Dirichlet-type constraint is applied on the complete boundary, also due to the simple boundary conditions. We also provide results for NL-BDDC without any interior update $\delta\bar{u}_I^{(k)}$ in the outer Newton iteration in Figure 4.9; see (2.17) and the discussion in the following lines. Omitting the update leads to an increased number of inner iterations and total time to solution. Therefore, we included the update in all other numerical results.

ETNA
Kent State University and
Johann Radon Institute (RICAM)

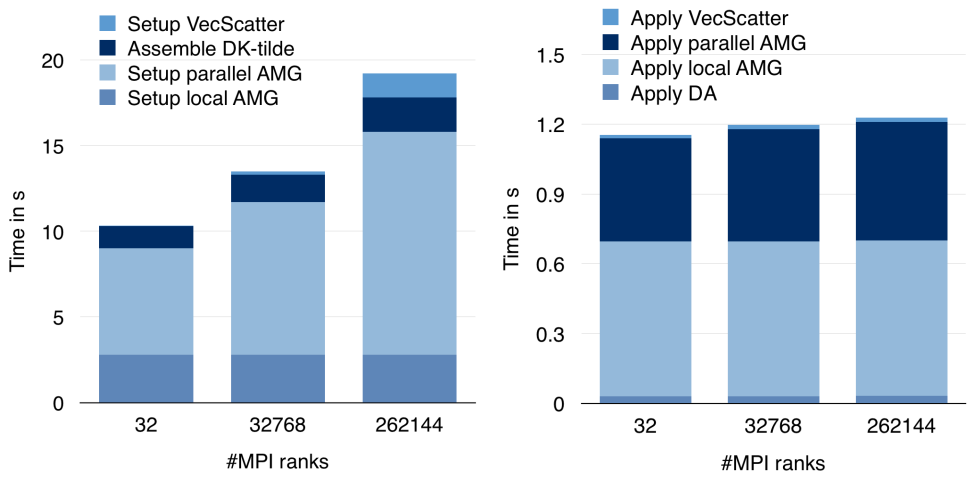

FIG. 4.7. *Weak scaling of* $\widehat{M}_2^{-1}$ *for a linear elasticity problem in three dimensions.* **Left:** *setup of the linear BDDC preconditioner split into the setup of the local AMGs* $D\widehat{K}_{II}^{-1}$*, the setup of the parallel AMG* $D\widehat{K}^{-1}$*, the assembly of* $D\widetilde{K}$*, and the setup of the different VecScatter.* **Right:** *average time per CG iteration split into the application of* DA*, the application of* $D\widehat{K}_{II}^{-1}$*, the application of* $D\widehat{K}^{-1}$*, and the application of the different VecScatter operations. Using two MPI ranks for each core.*

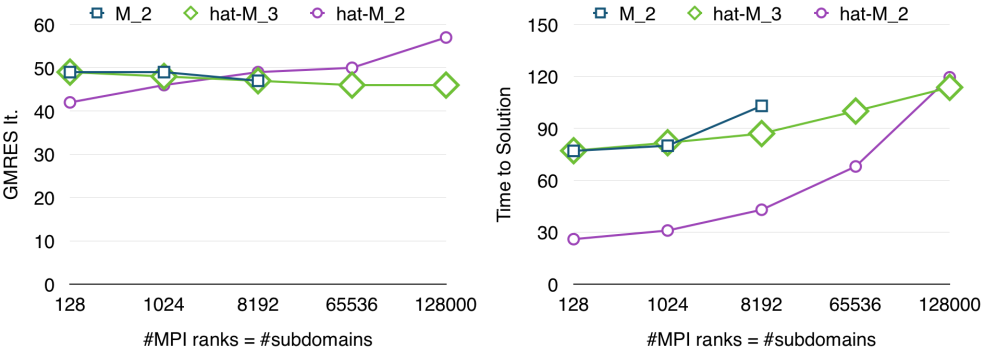

FIG. 4.8. **Linear 3D Cube** $\Omega = [0,2] \times [0,1] \times [0,1]$*; see Table 4.2 for a problem description.*

We perform similar experiments and obtain similar results in three dimensions; see Table 4.4. Here, NK-BDDC does not fail to converge for larger loads, but the number of GMRES iterations is drastically increased. In comparison, there is only a moderate increase in NL-BDDC.

For larger loads, NL-BDDC can be more than six times faster than NK-BDDC in three dimensions for a hyperelastic problems. We present the results with $4\,096$ MPI ranks and subdomains in Figure 4.12 and Figure 4.13.

We also compare the parallel efficiency of both methods in a weak scaling context. As a fair baseline, we always use the runtime of the faster of the two methods on 64 MPI ranks with the same load. Despite of nonlinear effects as, e.g., an increasing number of Newton steps, the parallel efficiency of NL-BDDC for $4\,096$ MPI ranks is higher than 72% for all investigated loads. In contrast, the parallel efficiency of NK-BDDC deteriorates drastically with an increasing load; see Figure 4.12 (right).

ETNA
Kent State University and
Johann Radon Institute (RICAM)

TABLE 4.3

Nonlinear 2D Square $\Omega = [0,1] \times [0,1]$*; heterogeneous and compressible Neo-Hooke material with* $E = 210\,000$*,* $\nu = 0.3$ *in circular inclusions inside the subdomains and* $E = 210$*,* $\nu = 0.3$ *in the remaining matrix material; P2 finite elements and* $H_s/h = 50$*; one V-cycle of BoomerAMG with nodal HMIS coarsening and extended+i interpolation is used for all AMG applications; for the parallel AMG the GM approach is used additionally; the BDDC coarse space consists of vertex constraints;* **It.** *denotes the number of GMRES iterations; computations for* $N = 64$ *and* $N = 1\,024$ *subdomains on* 2 *and* 32 *nodes of JUQUEEN, respectively; using two MPI ranks per computational core; discretized with* $P2$ *elements.*

N	d.o.f.	load	It.		Time to Sol.		Outer It.		Inner It.	
			NK	NL	NK	NL	NK	NL	NK	NL
64	1 283 202	0.01	125	72	83.5s	66.8s	3	2	-	5
		0.1	169	97	114.0s	91.8s	4	2	-	7
		0.2	293	101	190.1s	106.0s	5	3	-	9
		0.3	213	99	143.8s	104.4s	5	3	-	9
		0.4	345	130	228.1s	126.1s	5	3	-	9
		0.5	357	119	233.3s	121.0s	6	3	-	10
		0.6	-	139	-	182.6s	-	3	-	13
		0.7	-	143	-	146.1s	-	3	-	10
		0.8	-	139	-	170.7s	-	3	-	11
1 024	20 492 802	0.01	146	87	97.1s	76.6s	3	2	-	5
		0.1	201	116	134.1s	104.1s	4	2	-	7
		0.2	395	125	255.9s	114.7s	5	2	-	8
		0.3	317	116	208.0s	107.8s	5	2	-	8
		0.4	643	151	430.9s	170.7s	6	2	-	8
		0.5	621	132	396.0s	130.8s	6	3	-	10
		0.6	-	155	-	188.3s	-	3	-	13
		0.7	-	181	-	171.7s	-	3	-	10
		0.8	-	163	-	184.7s	-	3	-	11

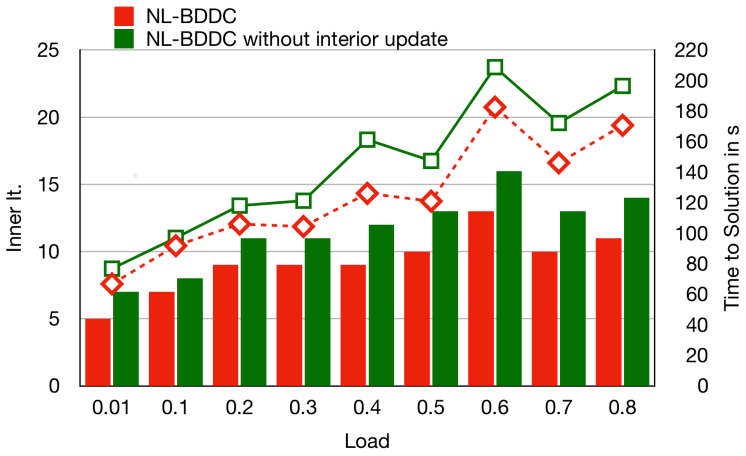

FIG. 4.9. *Increasing load for NL-BDDC with and without interior update in the outer loop for an inhomogeneous Neo-Hooke hyperelasticity problem in two dimensions and* 64 *subdomains; see Table 4.3 for the corresponding problem description. The bar plots belong to the inner iterations.*

ETNA
Kent State University and
Johann Radon Institute (RICAM)

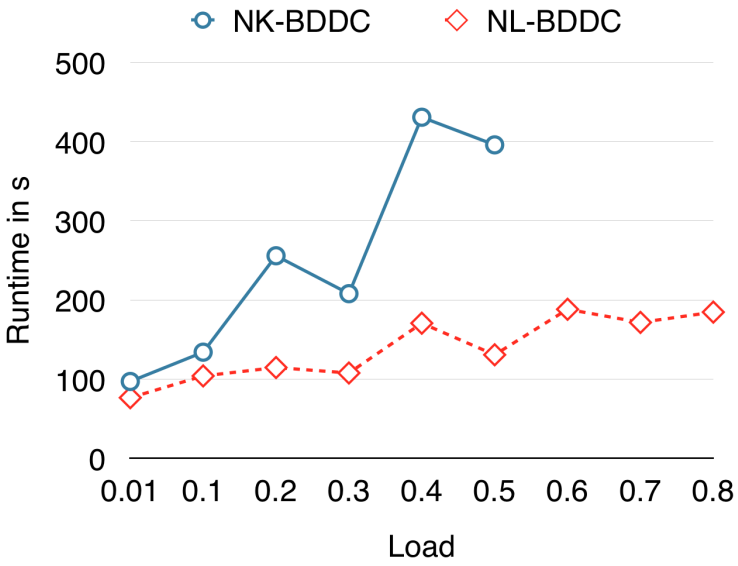

FIG. 4.10. **Time to Solution** *for an increasing load for NL-BDDC and NK-BDDC for an inhomogeneous Neo-Hooke hyperelasticity problem in two dimensions and* 1 024 *subdomains; see Table 4.3 for the corresponding results and details. For load steps larger than* 0.5*, we do not have convergence for NK-BDDC.*

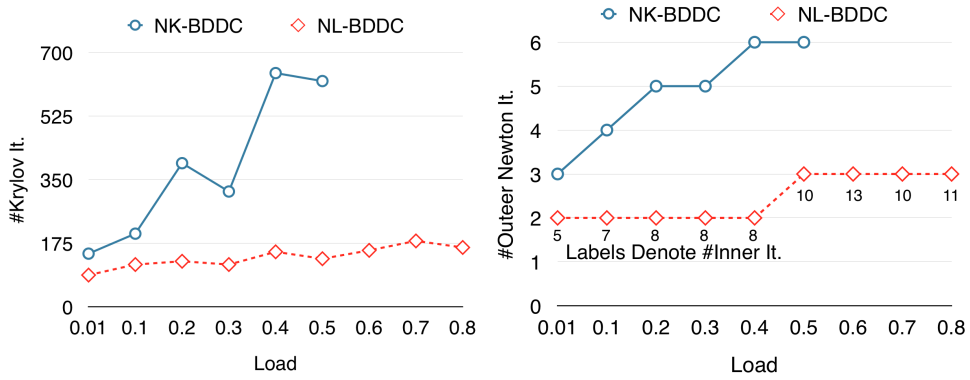

FIG. 4.11. *Increasing load for NL-BDDC and NK-BDDC for an inhomogeneous Neo-Hooke hyperelasticity problem in two dimensions and* 1 024 *subdomains;* **Left:** *total number of GMRES iterations (sum over all Newton steps);* **Right:** *number of Newton iterations, the number of inner Newton iterations in NL-BDDC is marked too; see the labels; see Table 4.3 for the corresponding results and details.*

For the case of load $= 0.8$, we also show the superior scalability of NL-BDDC by presenting a complete weak scaling study from 64 to 262 144 MPI ranks in Figure 4.14. For the 262 144 MPI rank case, Newton-Krylov did not converge in a reasonable runtime.

4.5. Nonlinear J2 elasto-plasticity with increasing load. Similar to Section 4.4, we apply the deformation

$$F = \begin{bmatrix} \text{load} & 0 & 0 \\ 0 & -\text{load} \cdot \nu & 0 \\ 0 & 0 & -\text{load} \cdot \nu \end{bmatrix}$$

ETNA
Kent State University and
Johann Radon Institute (RICAM)

TABLE 4.4
Nonlinear 3D Cube $\Omega = [0,1] \times [0,1] \times [0,1]$*; heterogeneous and compressible Neo-Hooke material with* $E = 210\,000$ *and* $\nu = 0.3$ *in inclusions inside subdomains and* $E = 210$ *and* $\nu = 0.3$ *in the remaining matrix material; one V-cycle of BoomerAMG with nodal HMIS coarsening and extended+i interpolation for all AMG applications; the BDDC coarse space consists of vertex constraints and the midpoints of all edges;* **It.** *denotes the number of GMRES iterations;* **Eff.** *denotes the parallel efficiency, with the faster of the two methods for the same load on* 64 *MPI ranks as a baseline; computations for* $N = 64$ *and* $N = 4\,096$ *subdomains on* 2 *and* 128 *nodes of JUQUEEN, respectively; using 2 MPI ranks per computational core; discretized with* $P1$ *elements.*

N	d.o.f.	load	It.		Time to Sol.		Outer It.		Inner It.		Eff.	
			NK	NL	NK	NL	NK!	NL	NK	NL	NK	NL
64	206 763	0.01	129	59	30.4s	29.2s	4	2	-	5	96%	100%
		0.2	219	91	52.7s	51.9s	7	3	-	10	98%	100%
		0.4	906	100	142.0s	64.8s	10	3	-	14	46%	100%
		0.6	296	91	63.6s	63.0s	8	3	-	14	99%	100%
		0.8	582	143	98.7s	74.7s	9	4	-	15	76%	100%
		1.0	2081	323	256.5s	111.0s	7	5	-	19	43%	100%
4 096	12 519 843	0.01	180	120	39.1s	40.8s	3	2	-	5	75%	72%
		0.2	441	128	95.9s	63.2s	6	3	-	10	54%	82%
		0.4	1087	150	216.0s	80.0s	10	3	-	14	30%	81%
		0.6	442	144	105.0s	78.4s	8	3	-	14	60%	80%
		0.8	869	226	170.6s	95.6s	9	4	-	15	44%	78%
		1.0	4751	538	747.3s	151.1s	15	5	-	17	15%	73%

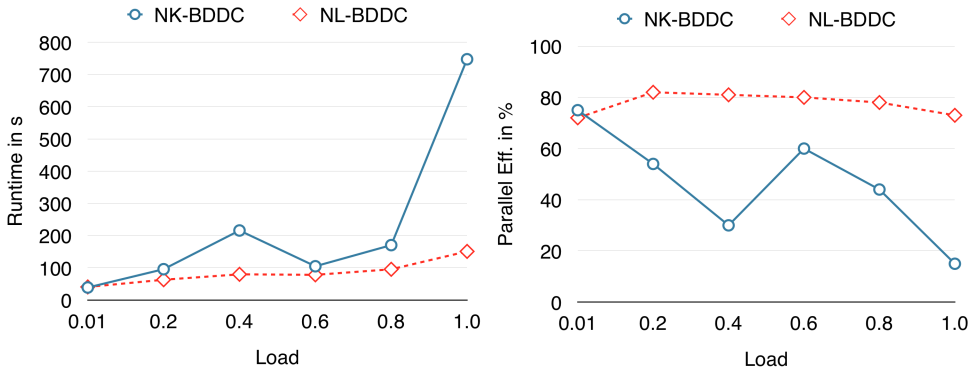

FIG. 4.12. **Left:** *time to solution for an increasing load for NL-BDDC and NK-BDDC for an inhomogeneous Neo-Hooke hyperelasticity problem in three dimensions and* 4 096 *subdomains;* **Right:** *parallel efficiency for the different loads; the runtime of the faster method for the same load on* 64 *MPI ranks is used as a baseline; see Table 4.4 for the corresponding results and details.*

with different choices for load to the three-dimensional cube $[0,1] \times [0,1] \times [0,1]$ built of dual-phase steel. We used a J2 elasto-plasticity model with an exponential hardening. The parameter choices and different yield stresses for the martensitic and ferritic phases can be found in the caption of Table 4.5 or, in more detail, in [7, Fig. 10]. In contrast to Section 4.4, we do not apply the complete load in a single load step but in several incremental load steps.

Considering local martensitic inclusions enclosed in a single subdomain each, in Table 4.5, NL-BDDC is again superior and more stable. Only for small loads (0.01) applied in many load steps (more than 2), NK-BDDC shows a better performance. For larger loads, especially if applied in few steps, NL-BDDC is faster. The load of 0.08 applied in 2 load steps can finally not be handled by NK-BDDC. In contrast, NL-BDDC shows a slight increase of 14% in runtime compared to applying half of the load, i.e., 0.04.

Finally, in Table 4.6, we consider a representative volume for a dual-phase steel with martensite and ferrite phases; see Figure 4.15. For this challenging problem, we see that for a

ETNA
Kent State University and
Johann Radon Institute (RICAM)

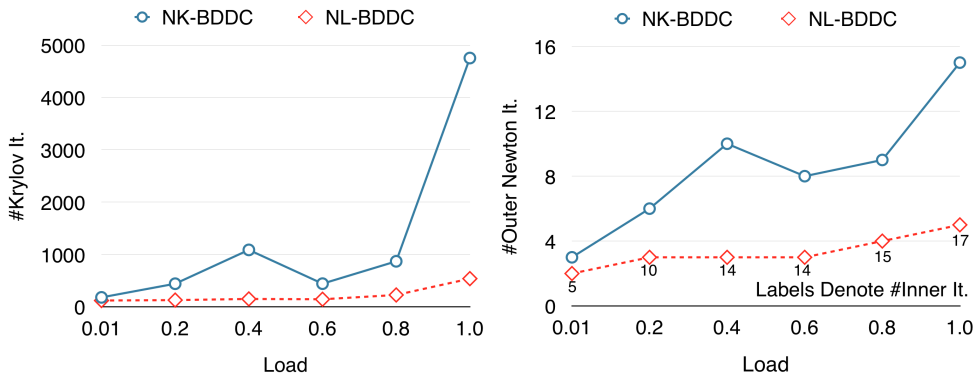

FIG. 4.13. *Increasing load for NL-BDDC and NK-BDDC for an inhomogeneous Neo-Hooke hyperelasticity problem in three dimensions and* 4 096 *subdomains;* **Left:** *total number of GMRES iterations (sum over all Newton steps);* **Right:** *number of Newton iterations, the number of inner Newton iterations in NL-BDDC is marked too; see the labels; see Table 4.4 for the corresponding results and details.*

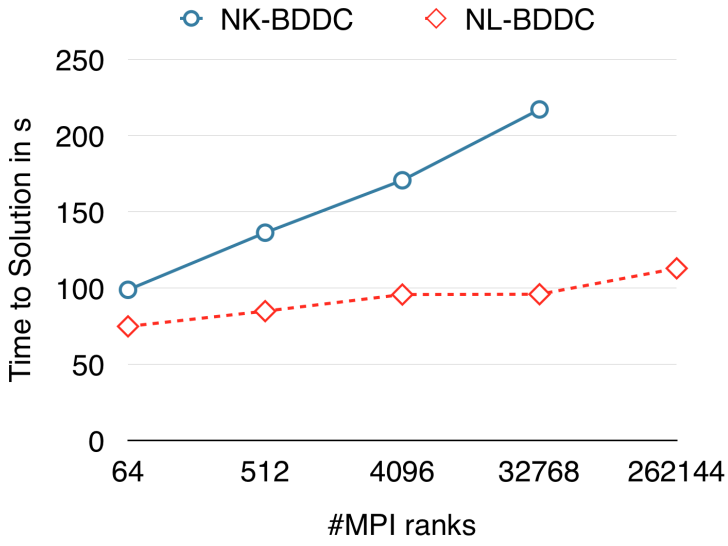

FIG. 4.14. *Weak scaling for a nonlinear Neo-Hooke problem; see Table 4.4 for a problem description. NK-BDDC failed to solve the problem on* 262 144 *in a reasonable time.*

small load of 0.005, the performance of NK and NL is very similar with a slight advantage for NK. However, for a load step of 0.01 and the largest load step, the NK approach breaks down whereas NL continues to converge.

Nevertheless, we again observe an enlarged convergence radius for large loads (0.01) applied in only two load steps, where NK-BDDC diverges and NL-BDDC remains stable.

5. Conclusion. We have presented a new and highly scalable nonlinear BDDC preconditioner with inexact solvers. For linear problems, the method reduces to the approach introduced in [36]. We have provided a discussion and comparison of different inexact and approximate BDDC approaches [16, 36] from the literature.

Weak parallel scalability of the new method and the implementation was shown to more than half a million parallel tasks for linear elasticity problems in two and three dimensions. We were able to show the improved performance of the new nonlinear BDDC method compared

ETNA
Kent State University and
Johann Radon Institute (RICAM)

TABLE 4.5
Nonlinear 3D Dual-Phase Steel Cube $\Omega = [0,1] \times [0,1] \times [0,1]$*; heterogeneous J2 elasto-plasticity material with exponential hardening,* $E = 206\,000$*, and* $\nu = 0.3$*; in the martensitic inclusions inside the subdomains the yield stresses are* $y_0 = 1\,000$ *and* $y_\infty = 2\,750$*; in the ferritic matrix material the yield stresses are* $y_0 = 260$ *and* $y_\infty = 580$*; one V-cycle of BoomerAMG with nodal HMIS coarsening and extended+i interpolation for all AMG applications; the LN approach is used for parallel AMG applications; the BDDC coarse space consists of vertex constraints and the midpoints of all edges;* **It.** *denotes the number of GMRES iterations; computations for* $N = 216$ *subdomains on* 216 *cores of MagnitUDE; discretized with piecewise quadratic elements (*$P2$*).*

Load	Load Steps	It.		Time to Sol.		Outer It.		Inner It.	
		NK	NL	NK	NL	NK	NL	NK	NL
0.01	8	596	538	226.3s	254.9s	22	21	-	28
	4	434	379	153.9s	162.3s	13	12	-	17
	2	457	230	152.6s	100.6s	10	6	-	10
0.02	8	1 058	865	374.8s	371.8s	29	24	-	37
	4	818	498	280.5s	220.0s	18	12	-	22
	2	458	326	160.6s	144.3s	9	6	-	12
0.04	8	1 534	1 049	531.2s	461.2s	34	24	-	45
	4	843	640	296.9s	284.4s	17	12	-	24
	2	852	470	296.1s	211.4s	12	8	-	14
0.08	8	1 594	1 251	561.6s	554.7s	33	24	-	48
	4	1 341	909	465.6s	402.9s	20	16	-	28
	2	-	540	-	241.9s	-	8	-	16

TABLE 4.6
Nonlinear 3D RVE $\Omega = [0,1] \times [0,1] \times [0,1]$*; same material as in Table 4.5; the material distribution, obtained from an EBSD measurement of dual-phase steel, is depicted in Figure 4.15; one V-cycle of BoomerAMG with nodal HMIS coarsening and extended+i interpolation for all AMG applications; the LN approach is used for parallel AMG applications; the BDDC coarse space consists of vertex constraints and the midpoints of all edges;* **It.** *denotes the number of GMRES iterations; computations for* $N = 216$ *subdomains on* 216 *cores of MAgnitUDE; discretized with piecewise quadratic elements (*$P2$*).*

Load	Load Steps	It.		Time to Sol.		Outer It.		Inner It.	
		NK	NL	NK	NL	NK	NL	NK	NL
0.005	8	502	466	187.9s	214.1s	20	19	-	25
	4	285	267	109.9s	123.2s	11	10	-	13
	2	238	216	82.5s	88.6s	7	6	-	9
0.01	8	762	693	288.2s	302.8s	27	22	-	29
	4	556	504	191.4s	213.5s	15	14	-	23
	2	-	466	-	167.2s	-	9	-	19

to the standard Newton-Krylov BDDC approach. Our computations show that NL-BDDC can deliver the result of a simulation up to six times faster than a classical Newton-Krylov approach and shows superior parallel scalability for nonlinear problems.

We also show that the performance on the Knights Landing supercomputer Theta can be more than two times better than that on the BG/Q supercomputer JUQUEEN using the same number of cores.

Acknowledgments. This work was supported in part by the German Research Foundation (DFG) through the Priority Programme 1648 "Software for Exascale Computing" (**SPPEXA**) under grants KL 2094/4-1, KL 2094/4-2, RH 122/2-1, and RH 122/3-2.

This research used resources of the Argonne Leadership Computing Facility (**Theta**), which is a DOE Office of Science User Facility supported under Contract DE-AC02-06CH11357,

ETNA
Kent State University and
Johann Radon Institute (RICAM)

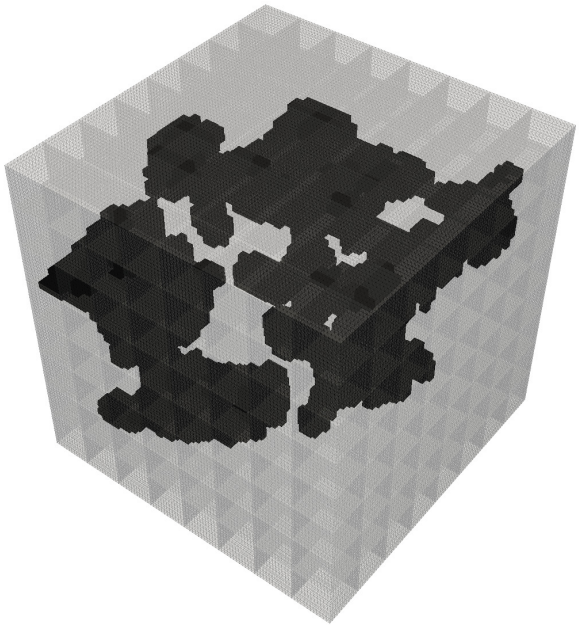

FIG. 4.15. *Material distribution of dual-phase steel obtained by an EBSD measurement used for computations in Table 4.6. This RVE is part of a larger structure presented in [7].*

within the Director's Discretionary Program.

The authors gratefully acknowledge the computing time granted by the Center for Computational Sciences and Simulation (CCSS) of the Universität 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).

The authors also gratefully acknowledge the Gauss Centre for Supercomputing e.V. (www.gauss-centre.eu) **JUQUEEN** [40] at Jülich Supercomputing Centre (JSC, www.fz-juelich.de/ias/jsc). GCS is the alliance of the three national supercomputing centres HLRS (Universität Stuttgart), JSC (Forschungszentrum Jülich), and LRZ (Bayerische Akademie der Wissenschaften), funded by the German Federal Ministry of Education and Research (BMBF) and the German State Ministries for Research of Baden-Württemberg (MWK), Bayern (StMWFK) and Nordrhein-Westfalen (MIWF).

REFERENCES

[1] S. BADIA, A. F. MARTÍN, AND J. PRINCIPE, *On the scalability of inexact balancing domain decomposition by constraints with overlapped coarse/fine corrections*, Parallel Comput., 50 (2015), pp. 1–24.

[2] ———, *Multilevel balancing domain decomposition at extreme scales*, SIAM J. Sci. Comput., 38 (2016), pp. C22–C52.

[3] A. H. BAKER, A. KLAWONN, T. KOLEV, M. LANSER, O. RHEINBACH, AND U. M. YANG, *Scalability of classical algebraic multigrid for elasticity to half a million parallel tasks*, in Software for Exascale Computing—SPPEXA 2013–2015, H.-J. Bungartz, P. Neumann, and E. W. Nagel, eds., vol. 113 of Lect. Notes Comput. Sci. Eng., Springer, Cham, 2016, pp. 113–140.

[4] A. H. BAKER, T. V. KOLEV, AND U. M. YANG, *Improving algebraic multigrid interpolation operators for linear elasticity problems*, Numer. Linear Algebra Appl., 17 (2010), pp. 495–517.

[5] S. BALAY, W. D. GROPP, L. C. MCINNES, AND B. F. SMITH, *Efficient management of parallelism in object oriented numerical software libraries*, in Modern Software Tools in Scientific Computing, E. Arge, A. M. Bruaset, and H. P. Langtangen, eds., Birkhäuser, Boston, 1997, pp. 163–202.

[6] F. BORDEU, P.-A. BOUCARD, AND P. GOSSELET, *Balancing domain decomposition with nonlinear relocal-*

ETNA
Kent State University and
Johann Radon Institute (RICAM)

ization: parallel implementation for laminates, in Proc. First Int. Conf. on Parallel, Distributed and Grid Computing for Engineering, B. H. V. Topping and P. Iványi, eds., Civil-Comp Press, Stirlingshire, 2009, Paper 4.

[7] D. BRANDS, D. BALZANI, L. SCHEUNEMANN, J. SCHRÖDER, H. RICHTER, AND D. RAABE, *Computational modeling of dual-phase steels based on representative three-dimensional microstructures obtained from ebsd data*, Arch. Appl. Mech., 86 (2016), pp. 575–598.

[8] P. R. BRUNE, M. G. KNEPLEY, B. F. SMITH, AND X. TU, *Composing scalable nonlinear algebraic solvers*, SIAM Rev., 57 (2015), pp. 535–565.

[9] X.-C. CAI AND D. E. KEYES, *Nonlinearly preconditioned inexact Newton algorithms*, SIAM J. Sci. Comput., 24 (2002), pp. 183–200.

[10] 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.

[11] J.-M. CROS, *A preconditioner for the Schur complement domain decomposition method*, in Domain Decomposition Methods in Science and Engineering, I. Herrera, D. Keyes, O. B. Widlund, and R. Yates, eds., National Autonomous University of Mexico (UNAM), Mexico City, 2003, pp. 373–380.

[12] T. A. DAVIS, *A column pre-ordering strategy for the unsymmetric-pattern multifrontal method*, ACM Trans. Math. Software, 30 (2004), pp. 167–195.

[13] H. DE STERCK, R. D. FALGOUT, J. W. NOLTING, AND U. M. YANG, *Distance-two interpolation for parallel algebraic multigrid*, Numer. Linear Algebra Appl., 15 (2008), pp. 115–139.

[14] H. DE STERCK, U. M. YANG, AND J. J. HEYS, *Reducing complexity in parallel algebraic multigrid preconditioners*, SIAM J. Matrix Anal. Appl., 27 (2006), pp. 1019–1039.

[15] C. R. DOHRMANN, *A preconditioner for substructuring based on constrained energy minimization*, SIAM J. Sci. Comput., 25 (2003), pp. 246–258.

[16] ———, *An approximate BDDC preconditioner*, Numer. Linear Algebra Appl., 14 (2007), pp. 149–168.

[17] V. DOLEAN, M. J. GANDER, W. KHERIJI, F. KWOK, AND R. MASSON, *Nonlinear preconditioning: how to use a nonlinear Schwarz method to precondition Newton's method*, SIAM J. Sci. Comput., 38 (2016), pp. A3357–A3380.

[18] C. FARHAT, M. LESOINNE, AND K. PIERSON, *A scalable dual-primal domain decomposition method*, Numer. Linear Algebra Appl., 7 (2000), pp. 687–714.

[19] C. GROSS, *A Unifying Theory for Nonlinear Additively and Multiplicatively Preconditioned Globalization Strategies: Convergence Results and Examples From the field of Nonlinear Elastostatics and Elastodynamics*, PhD. Thesis, Math-Nat. Fak., RFW University Bonn, Bonn, 2009.

[20] C. GROSS AND R. KRAUSE, *On the globalization of ASPIN employing trust-region control strategies - convergence analysis and numerical examples*, Tech. Rep. 2011-03, Inst. Comp. Sci., Universita della Svizzera italiana, Lugano, 2011.

[21] V. E. HENSON AND U. M. YANG, *BoomerAMG: a parallel algebraic multigrid solver and preconditioner*, Appl. Numer. Math., 41 (2002), pp. 155–177.

[22] F.-N. HWANG AND X.-C. CAI, *Improving robustness and parallel scalability of Newton method through nonlinear preconditioning*, in Domain Decomposition Methods in Science and Engineering, R. Kornhuber, R. Hoppe, J. Périaux, O. Pironneau, O. Widlund, and J. Xu, eds., vol. 40 of Lect. Notes Comput. Sci. Eng., Springer, Berlin, 2005, pp. 201–208.

[23] ———, *A class of parallel two-level nonlinear Schwarz preconditioned inexact Newton algorithms*, Comput. Methods Appl. Mech. Engrg., 196 (2007), pp. 1603–1611.

[24] A. KLAWONN, M. LANSER, AND O. RHEINBACH, *Nonlinear FETI-DP and BDDC methods*, SIAM J. Sci. Comput., 36 (2014), pp. A737–A765.

[25] ———, *FE2TI (ex_nl/fe^2) EXASTEEL - Bridging scales for multiphase steels*, Software, 2015. http://www.fz-juelich.de/ias/jsc/EN/Expertise/High-Q-Club/FE2TI/_node.html

[26] ———, *Toward extremely scalable nonlinear domain decomposition methods for elliptic partial differential equations*, SIAM J. Sci. Comput., 37 (2015), pp. C667–C696.

[27] ———, *FE^{2}TI: computational scale bridging for dual-phase steels*, in Parallel Computing: On the Road to Exascale; Proceedings of ParCo2015, G. R. Joubert, H. Leather, M. Parsons, F. Peters, and M. Sawyer, eds., IOS Series Advances in Parallel Computing, IOS Press, Amsterdam, 2016, pp. 797–806.
Also: TUBAF Preprint: 2015-12, TU Bergakademie Freiberg, Freiberg http://tu-freiberg.de/fakult1/forschung/preprints.

[28] ———, *A highly scalable implementation of inexact nonlinear FETI-DP without sparse direct solvers*, in Numerical mathematics and advanced applications—ENUMATH 2015, B. Karasözen, M. Manguoğlu, M. Tezer-Sezgin, S. Göktepe, and Ö. Uğur, eds., vol. 112 of Lect. Notes Comput. Sci. Eng., Springer, Cham, 2016, pp. 255–264.

[29] ———, *Using algebraic multigrid in inexact BDDC domain decomposition methods*, in Domain Decomposition Methods in Science and Exngineering XXIV, P. E. Bjørstad, S. C. Brenner, L. Halpern, R. Kornhuber,

ETNA
Kent State University and
Johann Radon Institute (RICAM)

H. H. Kim, T. Rahmann, O. B. Widlund, eds., in vol. 125 of Lect. Notes Comput. Sci. Eng., Springer, Basel, 2017, in press.

[30] A. KLAWONN, M. LANSER, O. RHEINBACH, AND M. URAN, *Nonlinear FETI-DP and BDDC methods: a unified framework and parallel results*, SIAM J. Sci. Comput., 39 (2017), pp. C417–C451.

[31] A. KLAWONN, L. F. PAVARINO, AND O. RHEINBACH, *Spectral element FETI-DP and BDDC preconditioners with multi-element subdomains*, Comput. Methods Appl. Mech. Engrg., 198 (2008), pp. 511–523.

[32] A. KLAWONN AND O. RHEINBACH, *Inexact FETI-DP methods*, Internat. J. Numer. Methods Engrg., 69 (2007), pp. 284–307.

[33] ———, *Highly scalable parallel domain decomposition methods with an application to biomechanics*, ZAMM Z. Angew. Math. Mech., 90 (2010), pp. 5–32.

[34] J. LI AND O. B. WIDLUND, *On the use of inexact subdomain solvers for BDDC algorithms*, Tech. Rep. TR2005-871, Department of Computer Science, Courant Institute, New York, 2005.

[35] ———, *FETI-DP, BDDC, and block Cholesky methods*, Internat. J. Numer. Methods Engrg., 66 (2006), pp. 250–271.

[36] ———, *On the use of inexact subdomain solvers for BDDC algorithms*, Comput. Methods Appl. Mech. Engrg., 196 (2007), pp. 1415–1428.

[37] 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.

[38] J. MANDEL, B. SOUSEDÍK, AND C. R. DOHRMANN, *Multispace and multilevel BDDC*, Computing, 83 (2008), pp. 55–85.

[39] J. PEBREL, C. REY, AND P. GOSSELET, *A nonlinear dual-domain decomposition method: application to structural problems with damage*, Inter. J. Multiscal Comp. Eng., 6 (2008), pp. 251–262.

[40] M. STEPHAN AND J. DOCTER, *JUQUEEN: IBM Blue Gene/Q® Supercomputer System at the Jülich Supercomputing Centre*, Journal Large-Scale Res. Facilities, 1 (2015), Article A1, 18 pages.

[41] A. TOSELLI AND O. B. WIDLUND, *Domain Decomposition Methods—Algorithms and Theory*, Springer, Berlin, 2005.

[42] X. TU, *Three-level BDDC in three dimensions*, SIAM J. Sci. Comput., 29 (2007), pp. 1759–1780.

[43] U. M. YANG, *On long-range interpolation operators for aggressive coarsening*, Numer. Linear Algebra Appl., 17 (2010), pp. 453–472.

[44] S. ZAMPINI, *PCBDDC: a class of robust dual-primal methods in PETSc*, SIAM J. Sci. Comput., 38 (2016), pp. S282–S306.