Preprint, Institute of Mathematics, AS CR, Prague. 2009-11-2

Adaptive BDDC in Three Dimensions

Jan Mandel, Bedřich Sousedík and Jakub Šístek *

Abstract

The adaptive BDDC method is extended to the selection of face constraints in three dimensions. A new implementation of the BDDC method is presented based on a global formulation without an explicit coarse problem, with massive parallelism provided by a multifrontal solver. Constraints are implemented by a projection and sparsity of the projected operator is preserved by a generalized change of variables. The effectiveness of the method is illustrated on several engineering problems.

1 Introduction

The *Balancing Domain Decomposition by Constraints* (BDDC) was developed by Dohrmann [6] as a primal alternative to the *Finite Element Tearing and Interconnecting - Dual, Primal* (FETI-DP) by Farhat et al. [7]. Both methods use constraints to impose equality of new "coarse" variables on substructure interfaces, such as values at substructure corners or weighted averages over edges and faces. Primal variants of the FETI-DP were also independently proposed by Cros [4] and by Fragakis and Papadrakakis [10]. It has been shown in [28, 38] that these methods are in fact the same as BDDC. Polylogarithmic condition number bounds for FETI-DP were first proved in [30] and generalized to the case of coefficient jumps between substructures in [14]. The same bounds were obtained for BDDC in [24, 25]. A proof that the eigenvalues of the preconditioned operators of both methods are actually the same except for the eigenvalues equal to one was given in [25] and then simplified in [2, 20, 28]. FETI-DP, and, equivalently, BDDC are quite robust. It can be proved that the condition number remains bounded even for large classes of subdomains with rough interfaces in 2D [12, 40] as well as in many cases of strong discontinuities of coefficients, including some configurations when the discontinuities cross substructure boundaries [32, 33]. However, the condition number does deteriorate in many situations of practical importance and an adaptive method is warranted. Enriching the coarse space so that the iterations run in a subspace devoid of "difficult" modes has been a long-standing trick in iterative substructuring methods and used, e.g., in the development of BDD and FETI for plates from the base BDD and FETI methods [8, 18, 19, 31]. Methods that build a coarse space

*Corresponding author at: Institute of Mathematics, Academy of Sciences of the Czech Republic, Žitná 25, CZ–115 67 Praha 1, Czech Republic. Tel.: +420 222 090 710; fax: +420 222 211 638, sistek@math.cas.cz.

adaptively from local eigenvalue calculations were also devised in a number of other contexts [3, 9, 21, 22, 34]. Adaptive enrichment for BDDC and FETI-DP was proposed in [26, 27], with the added coarse functions built from eigenproblems based on adjacent pairs of substructures in 2D. The adaptive method, however, was formulated in terms of FETI-DP operators, and it was quite complicated.

Here, we develop the adaptive algorithm directly in terms of BDDC operators, resulting in a much simplified formulation and implementation. Of course, the algorithm still allows a translation into the language of the FETI-DP. We then extend the construction from [26, 27] to 3D. We find that the heuristic eigenvalue-based estimates still work reasonably well and that our adaptive approach can result in the concentration of computational work in a small troublesome part of the problem, which leads to a good convergence behaviour at a small added cost.

We also develop a new implementation framework that operates on global matrices, builds no explicit coarse problem, and gets much of its parallelism through the direct solver used for solution of an auxiliary decoupled system. To preserve sparsity, we use a variant of the change of variables from [20], extended to an arbitrary number of constraints. Our current parallel implementation is built on top of the multifrontal massively parallel sparse direct solver MUMPS [1], motivated also by an earlier implementation of the BDDC preconditioner based on the frontal solver [35].

The rest of the paper is organized as follows. In Section 2, we establish the notation and review the BDDC algorithm in a form suitable for our purposes. In Section 3, we describe the adaptive method. Section 4 then describes the implementation on top of a massively parallel direct solver. Section 5 presents the generalized change of variables to preserve sparsity. Section 6 describes some further details of the implementation. Numerical results are presented in Section 7. Section 8 contains the summary and concluding remarks.

Some of results in this paper were presented in the thesis [37].

2 Notation, substructuring, and BDDC

To establish notation, we first briefly review standard substructuring concepts and state the BDDC method in a form suitable for our purposes. The setting and notation here is compatible with [29], with some additions. See, e.g., [36, 39] for more details about iterative substructuring and [16, 24, 29, 36, 39] for BDDC.

Consider a bounded domain $\Omega \subset \mathbb{R}^3$ discretized by conforming finite elements. The domain Ω is decomposed into N nonoverlapping *subdomains* Ω_i, $i = 1, \dots N$, also called *substructures*, so that each substructure Ω_i is a union of finite elements. Each node is associated with one degree of freedom in the scalar case, and with 3 displacement degrees of freedom in the case of linear elasticity. The nodes contained in the intersection of at least two substructures are called boundary nodes. The union of all boundary nodes of all substructures is called the *interface*, denoted by Γ, and Γ_i is the interface of substructure Ω_i. The interface Γ may also be classified as the union of three different types of sets: *faces*, *edges* and *corners*. We will adopt here a simple (geometric) definition: a *face* contains all

nodes shared by the same two subdomains, an *edge* contains all nodes shared by same set of more than two subdomains, and a *corner* is a degenerate edge with only one node; for a more general definition see, e.g., [13]. Similarly as in [23], edges, and faces are called *globs*.

We identify finite element functions with the vectors of their coefficients in the standard finite element basis. These coefficients are also called *variables* or *degrees of freedom*. We also identify linear operators with their matrices, in bases that will be clear from the context.

The space of all (vectors of the degrees of freedom of) finite element functions on subdomain Ω_i is denoted by W_i, and let

$$W = W_1 \times \cdots \times W_N. \tag{1}$$

The space W is equipped with the standard $\mathbb{R}^n$ basis.

The Euclidean inner product $\langle w, v\rangle = w^{\mathrm{T}} v$. For a symmetric positive semidefinite matrix M, $\langle u, v\rangle_M = \langle Mu, v\rangle$, and $\|u\|_M = \langle Mu, u\rangle^{1/2}$.

Let $A_i : W_i \to W_i$ be the local substructure stiffness matrix, obtained by the subassembly of element matrices only in substructure Ω_i. The matrices A_i are symmetric positive semidefinite. We can write vectors and matrices in the block form

$$w = \begin{bmatrix} w_1 \\ \vdots \\ w_N \end{bmatrix}, \quad w \in W, \quad A = \begin{bmatrix} A_1 & & \\ & \ddots & \\ & & A_N \end{bmatrix} : W \to W. \tag{2}$$

Now let $U \subset W$ be the space of all functions from W that are continuous across substructure interfaces. We are interested in solving the problem

$$u \in U : \quad \langle Au, v\rangle = \langle f, v\rangle \quad \forall\, v \in U\,, \tag{3}$$

where $f \in W$ is a given right-hand side. Vectors from U are called vectors of *global* degrees of freedom, while vectors from W_i are called *local*. The space U is equipped with the basis of 0-1 vectors with one basis vector for each global degree of freedom. The basis vectors have 1s in the places where the global degree of freedom coincides with a local one. The matrix

$$R : U \to W,$$

formed from these basis vectors as columns, is the familiar global-to-local mapping that restricts the global vectors of degrees of freedom to local degrees of freedom on each Ω_i. Thus, $R^{\mathrm{T}} A R$ is the global stiffness matrix, and (3) is equivalent to the assembled system $R^{\mathrm{T}} A R v = R^{\mathrm{T}} f$. The matrix R is also the matrix of the canonical embedding $U \subset W$ in the given bases.

Denote by $U_I \subset W$ the space of all (vectors of) finite element functions with nonzero values only in the interiors of the substructures Ω_i. Then $U_I \subset U$, and the space W is decomposed as the A-orthogonal direct sum

$$W = U_I \oplus W_H, \quad U_I \perp_A W_H, \tag{4}$$

where the functions from W_H are called *discrete harmonic*. Such functions are fully determined by the values of the degrees of freedom at the boundaries of the substructures, and they have minimal energy on every subdomain. Therefore, in a computer implementation, *only the boundary values of discrete harmonic functions need to be stored*.

The A-orthogonal projection onto U_I is denoted by

$$P : W \to U_I.$$

For $w \in W$, $(I - P)\, w$ is the discrete harmonic extension from the values of w on the substructure boundaries. The evaluation of Pw consists of the solution of N independent Dirichlet problems, one in each substructure.

The space of all discrete harmonic functions from W that are continuous across the substructure interfaces is denoted by $\widehat{W}$. We have

$$\widehat{W} = W_H \cap U = (I - P)\, U,$$

and the A-orthogonal decomposition

$$U = U_I \oplus \widehat{W}, \quad U_I \perp_A \widehat{W}. \tag{5}$$

The solution $v \in \operatorname{Range} R$ is split as

$$v = u + w, \quad u \in U_I,\ w \in \widehat{W}.$$

Solving for the interior component $u \in U_I$ decomposes into N independent Dirichlet problems. We are interested in finding the discrete harmonic component $w \in \widehat{W}$, which is the solution of the reduced problem

$$w \in \widehat{W} : \quad \langle Aw, z \rangle = \langle f, z \rangle \quad \forall z \in \widehat{W}. \tag{6}$$

We require also an averaging operator

$$E : W \to U. \tag{7}$$

The operator E replaces the variables on the interfaces between the substructures by their averages from all adjacent subdomains, and it preserves variables in the interiors of the substructures. The operator E is a projection in W onto U. Then the operator

$$(I - P)\, E : W \to \widehat{W} \tag{8}$$

is a projection in W onto $\widehat{W}$. Its evaluation consists of averaging between the substructures, followed by the discrete harmonic extension from the substructure boundaries. Also, note that

$$(I - (I - P)\, E)\, w = (I - P)\, (I - E)\, w \quad \forall w \in W_H, \tag{9}$$

since $Pw = 0$ if $w \in W_H$.

Proper weights (e.g., proportional to the substructure stiffness) in the averaging by E are important for the performance of BDDC (as well as other iterative substructuring methods) independent of different stiffness of substructures [13, 25].

The BDDC preconditioner is characterized by a selection of *coarse degrees of freedom,* such as values at corners and averages over edges or faces. The action of the BDDC preconditioner is then defined in the space given by the requirement that the coarse degrees of freedom on adjacent substructures coincide, which is enforced in the algorithms by *constraints*. So, the design of the BDDC preconditioner is characterized by a selection of an intermediate space $\widetilde{W}$ satisfying these constraints,

$$\widehat{W} \subset \widetilde{W} \subset W_H. \tag{10}$$

The BDDC then consists of preconditioned conjugate gradients (PCG) applied to the problem (6) with the preconditioner

$$M_{BDDC} : r \mapsto u = (I - P) Ew, \quad w \in \widetilde{W} : \quad \langle Aw, z\rangle = \langle r, (I - P) Ez\rangle , \quad \forall z \in \widetilde{W}, \tag{11}$$

where r is the residual in the PCG method. The following condition number bound for BDDC will play an essential role in our design of the adaptive method.

Theorem 1 ([25]) *The eigenvalues of the preconditioned operator of the BDDC method satisfy* $1 \leq \lambda \leq \omega_{BDDC}$*, where*

$$\omega_{BDDC} = \sup_{w \in \widetilde{W}} \frac{\|(I - (I - P) E) w\|_A^2}{\|w\|_A^2}. \tag{12}$$

BDDC enforces the equality of corner coarse degrees of freedom directly by using the space W^c, consisting of all functions where the local degrees of freedom on the substructure corners coincide. Then

$$U \subset W^c \subset W.$$

Just like U, space W^c is equipped with a basis consisting of 0-1vectors. The basis vector corresponding to a corner degree of freedom has 1s in the places where the global degree of freedom coincides with the corresponding substructure degree of freedom. The global-to-local matrix

$$R^c : W^c \to W,$$

formed from these basis vectors as columns, is the matrix of the canonical embedding $W^c \subset W$, and

$$A^c = R^{c\mathrm{T}} A R^c. \tag{13}$$

is the stiffness matrix assembled at the subdomain corners only (Figure 1).

We assume that there are sufficiently many of corner constraints:

Assumption 1 *The matrix A is positive definite on W^c.*

Denote by $\widetilde{W^c}$ the space all of discrete harmonic functions in W^c. Then,

$$\widehat{W} \subset \widetilde{W} \subset \widetilde{W^c} = W^c \cap W_H \subset W_H, \tag{14}$$

and we construct the space $\widetilde{W}$ by enforcing the remaining constraints weakly,

$$\widetilde{W} = \left\{ w \in \widetilde{W^c} : Dw = 0 \right\}. \tag{15}$$

Each row of matrix D defines one constraint. We require that the constraints are satisfied by all functions that are continuous across the interfaces,

$$Dw = 0, \quad \forall w \in U. \tag{16}$$

Note that (16) implies that $\widehat{W} \subset \widetilde{W}$, and that the constraints $Dw = 0$ involve boundary variables only. The adaptive algorithm will construct such matrix D.

Lemma 2 *The BDDC preconditioner (11) satisfies*

$$M_{BDDC} : r \mapsto u = (I - P) E R^c w_c, \tag{17}$$

where for some λ,

$$\begin{array}{rcccc} A^c w_c & + & D^{c\mathrm{T}} \lambda & = & R^{c\mathrm{T}} E^{\mathrm{T}} (I - P)^{\mathrm{T}} r, \\ D^c w_c & & & = & 0. \end{array} \tag{18}$$

where

$$D^c = D R^c \tag{19}$$

differs from D *only by omitting some zero columns corresponding to corners.*

Proof. The saddle point problem (18) is equivalent to the constrained minimization,

$$\frac{1}{2} \langle Aw, w \rangle - \langle r, (I - P) Ew \rangle \to \min \quad \text{subject to } w \in W^c, Dw = 0, \tag{20}$$

with $w = R^c w_c$. Let w be a solution of (20) and $z \in U_I$. Since w is optimal with respect to variation z, and $(I - P) z = 0$, we have $\langle Aw, z \rangle = 0$. Thus, w is discrete harmonic. It follows (20) is equivalent to

$$\frac{1}{2} \langle Aw, w \rangle - \langle r, (I - P) Ew \rangle \to \min \quad \text{subject to } w \in \widetilde{W^c}, Dw = 0, \tag{21}$$

which, by (15), is the same as (11).

Finally, matrix DR^c differs from D only by omitting zero columns because the constraints $Dw = 0$ do not involve corners. ■

Remark 1 *In practice, the computation of* $(I - P)^{\mathrm{T}} r$ *can be omitted, because* $r = Ae$*, where the error* e *is discrete harmonic, and then*

$$\left\langle P^{\mathrm{T}} r, z \right\rangle = \left\langle P^{\mathrm{T}} A e, z \right\rangle = \langle Ae, Pz \rangle = \langle r, Pz \rangle = 0 \quad \forall z \in U_I,$$

thus $P^{\mathrm{T}} r = 0$*, that is,* $r = 0$ *in the interiors. The condition that the error* e *is discrete harmonic is preserved in the iteration by induction, and the initial error can be made discrete harmonic by a suitable choice of initial approximation for the reduced problem (e.g., zero).*

3 Adaptive selection of constraints

We first briefly review the principle of the adaptive method from [27], in a form suitable for our purposes. The condition number bound from Theorem 1 equals to the maximum eigenvalue λ_1 of the associated generalized eigenvalue problem

$$w \in \widetilde{W}: \quad \langle (I-(I-P)E)\,w, (I-(I-P)E)\,z\rangle_A = \lambda \langle w,z\rangle_A \quad \forall z \in \widetilde{W}. \tag{22}$$

The following statement is well known from linear algebra, e.g. [5, Theorem 5.2].

Lemma 3 (Courant-Fisher-Weyl minmax principle) *Let $d(\cdot,\cdot)$ be symmetric positive semidefinite bilinear form on vector space V of dimension n and $e(\cdot,\cdot)$ symmetric positive definite bilinear form on V. Then the generalized eigenvalue problem*

$$w \in V : c(w,u) = \lambda b(w,u) \quad \forall u \in V,$$

has n linearly independent eigenvectors w_k and the corresponding eigenvalues are real and nonnegative and the eigenvectors are stationary points of the Rayleigh quotient $c(w,w)/b(w,w)$, with the stationary values equal to λ_i. Order $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n \geq 0$. Then, for any subspace $V_k \subset V$ of dimension $n-k$,

$$\max_{w \in V_k, w \neq 0} \frac{c(w,w)}{b(w,w)} \geq \lambda_{k+1},$$

with equality if

$$V_k = \{w \in V : d(w_\ell, w) = 0, \quad \forall \ell = 1,\ldots,k\}. \tag{23}$$

Since the bilinear form on the left-hand side of (22) is symmetric positive semidefinite and the bilinear form on the right-hand side is symmetric positive definite, Lemma 3 applies:

Corollary 4 ([27]) *The generalized eigenvalue problem (22) has eigenvalues $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n \geq 0$. Denote the corresponding eigenvectors w_ℓ. Then, for any $k = 1,\ldots,n-1$, and any linear functionals L_ℓ on W, $\ell = 1,\ldots,k$,*

$$\max\left\{ \frac{\|(I-(I-P)E)\,w\|_A^2}{\|w\|_A^2} : w \in \widetilde{W},\ L_\ell(w) = 0\ \forall \ell = 1,\ldots,k \right\} \geq \lambda_{k+1},$$

with equality if

$$L_\ell(w) = \langle (I-(I-P)E)\,w_\ell, (I-(I-P)E)\,w\rangle_A. \tag{24}$$

Therefore, the optimal decrease of the condition number bound (12) can be achieved by adding to the constraint matrix D in the definition of $\widetilde{W}$ the rows d_ℓ defined by $d_\ell^{\mathrm{T}} w = L_\ell(w)$. However, solving the global eigenvalue problem (22) is expensive, and the vectors d_ℓ are not of the form required for substructuring, i.e., each d_ℓ with nonzeros in one glob only.

For these reasons, we replace (22) by a collection of local problems, each defined by considering only two adjacent subdomains Ω_i and Ω_j. Here, subdomains are considered adjacent if they share an edge in 2D, or a face in 3D (Figure 2). All quantities associated with such pair will be denoted by the subscript $_{ij}$. Using also (9), the generalized eigenvalue problem (24) becomes

$$w \in \widetilde{W}_{ij}: \quad \langle (I-P_{ij})(I-E_{ij})\, w, (I-P_{ij})(I-E_{ij})\, z\rangle_{A_{ij}} = \lambda \langle w, z\rangle_{A_{ij}} \quad \forall z \in \widetilde{W}_{ij}. \quad (25)$$

Assumption 2 *The corner constraints are already sufficient to prevent relative rigid body motions of any pair of adjacent substructures, so*

$$\forall w \in \widetilde{W}_{ij} : A_{ij} w = 0 \Rightarrow (I - E_{ij})\, w = 0,$$

i.e., the corner degrees of freedom are sufficient to constrain the rigid body modes of the two substructures into a single set of rigid body modes, which are continuous across the interface Γ_{ij}.

The maximal eigenvalue ω_{ij} of (25) is finite due to Assumption 2, and we define the heuristic *condition number indicator*

$$\widetilde{\omega} = \max \{\omega_{ij} : \Omega_i \text{ and } \Omega_j \text{ are adjacent}\}. \quad (26)$$

Considering two adjacent subdomains Ω_i and Ω_j only, we get the added constraints $L_\ell(w) = 0$ from (24) as

$$\langle (I-P_{ij})(I-E_{ij})\, w_\ell, (I-P_{ij})(I-E_{ij})\, w\rangle_{A_{ij}} = 0 \quad \forall \ell = 1, \ldots, k, \quad (27)$$

where w_ℓ are the eigenvectors corresponding to the k largest eigenvalues from (25).

To avoid complicated notation, we now drop the subscripts $_{ij}$, or, equivalently, consider a domain which consists of only two substructures.

Algorithm 1 (Adaptive BDDC [27]) *Find the smallest k for every two adjacent substructures to guarantee that $\lambda_{k+1} \le \tau$, where τ is a given tolerance, and add the constraints (27) to the definition of $\widetilde{W}$.*

The adaptive BDDC method assures that the condition number indicator $\widetilde{\omega} \le \tau$ with the minimum number of added constraints. It was presented in [27] starting from corner constraints only, formulated in terms of FETI-DP, and the result translated to BDDC. We extend the method to the case a general space $\widetilde{W}$ and give a much simpler implementation in BDDC directly.

The next lemma shows that the added constraints $L_\ell(w) = 0$ satisfy the compatibility condition (16).

Lemma 5 *The constraints $L_\ell(w) = 0$, with L_ℓ given by (24), are satisfied for any $w \in U$.*

Proof. From (9), $(I-(I-P)E) = (I-P)(I-E)$. For any $w \in U$, $(I-E)w = 0$, because E is a projection on U. ■

To formulate a numerical algorithm, we need to write the generalized eigenvalue problem (25) and the added constraints (27) in terms of matrices and vectors. Consider the space $\widetilde{W}$ given by the corner constraints and some initial constraint matrix D. Recall that W^c is the space of functions continuous at corners into W with the basis consisting of 0-1vectors such that the basis vector corresponding to a degree of freedom has 1 in the places where the global degree of freedom coincides with the corresponding substructure degree of freedom, $R^c : W^c \to W$ is the identity embedding so the matrix of R^c has the basis vectors of W^c as columns, $A^c = R^{c\mathrm{T}} A R^c$ is the matrix assembled at the corners, and $D^c = DR^c$, and $\Pi = I - D^{c\mathrm{T}} \left(D^c D^{c\mathrm{T}}\right)^{-1} D^c$ is the orthogonal projection onto $\operatorname{null} D^c$. The initial constraint matrix D can be empty; then $\Pi = I$. The generalized eigenvalue problem (22) now becomes

$$\Pi (I-E)^{\mathrm{T}} S^c (I-E) \Pi w = \lambda \Pi S^c \Pi w. \tag{28}$$

where

$$S^c = (I-P)^{\mathrm{T}} A^c (I-P).$$

Since

$$\operatorname{null} \Pi S^c \Pi \subset \operatorname{null} \Pi (I-E)^{\mathrm{T}} S^c (I-E) \Pi, \tag{29}$$

the eigenvalue problem (28) reduces in the factorspace modulo $\operatorname{null} \Pi S^c \Pi$ to the problem with the operator on the right-hand side positive definite. In some of the computations, we have used the subspace iteration method LOBPCG [15] to find the dominant eigenvalues and their eigenvectors. The LOBPCG iterations then simply run in the factorspace. To use standard eigenvalue solvers, (28) is converted to a matrix eigenvalue problem by penalizing the components in $\operatorname{null} D^c$ and rigid body modes, as already described in [27].

From the matrix form (29) of the eigenvalue problem, the constraints to be added are

$$L_\ell(w) = w_\ell^{\mathrm{T}} \Pi (I-E)^{\mathrm{T}} S^c (I-E) \Pi w = 0.$$

That is, we wish to add to the constraint matrix D the rows

$$d_\ell = w_\ell^{\mathrm{T}} \Pi (I-E)^{\mathrm{T}} S^c (I-E) \Pi. \tag{30}$$

Proposition 6 *The vectors d_ℓ, constructed for a domain consisting of only two substructures Ω_i and Ω_j, have matching entries on the interface between the two substructures, with opposite signs.*

Proof. Consider the vector $w \in W$ that has two entries equal to 1, corresponding to a degree of freedom on the interface, and all other entries equal to 0. Using the definition of d_ℓ and Lemma 5, we get $d_\ell w = L_\ell(w) = 0$. ■

In 2D, one can simply add rows (30) to the constraint matrix D, which is equivalent to the method from [27]. In 3D, unfortunately, the added rows will generally have nonzero entries over all of the interface of Ω_i and Ω_j, including the edges.(where Ω_i and Ω_j intersect other

substructures). Consequently, the added d_ℓ are not of the form required for substructuring, i.e., each d_ℓ with nonzeros in one glob only, and they will couple the globs together: the matrix DD^{T} is in general no longer block diagonal with one block per glob. To preserve the block diagonal structure, we have to split each d_ℓ into one row that contains the nonzero entries of the face, and one row for each edge that contains the nonzero entries of that edge. From Proposition 6, it follows that these split constraints satisfy the compatibility condition (16), and thus the space $\widetilde{W}$ is well defined.

Remark 2 *In the computations reported in Section 7, we drop the adaptively generated edge constraints in 3D. Then it is no longer guaranteed that the condition number indicator* $\widetilde{\omega} \leq \tau$. *However, the method is still observed to perform well.*

4 Parallel framework with global matrices and on top of multifrontal solver

The main purpose of BDDC, just like any other iterative substructuring method, is to split the problem into independent subproblems, which are solved independently on separate nodes in a multiprocessor system. Therefore, the usual implementation results in independent local problems on the spaces W_i and a small coarse problem [6, 24]. Parallel implementation then requires a fair amount of custom coding. To reduce the amount of new code, a BDDC implementation was developed [35] that uses specially crafted calls to a frontal solver to compute almost all quantities on the substructures. However, the frontal solver implementation needs to construct a coarse problem (which is solved by the same frontal solver), and the programmer needs to handle the parallelism explicitly. Fortunately, highly efficient massively parallel direct solvers exist, and an implementation based on such solver may avoid dealing with parallel issues completely. When there are only corner constraints, i.e., D is empty, the BDDC preconditioner (17)–(19) reduces to

$$M_{BDDC} : r \mapsto (I - P)\, E R^c\, {A^c}^{-1}\, {R^c}^{\mathrm{T}} E^T\, (I - P)^{\mathrm{T}}\, r. \tag{31}$$

All coupling in the matrix A^c between the substructures is concentrated at the corner degrees of freedom, while most computation work rests inside the subdomains, which an efficient solver should be able to perform independently in parallel. Our implementation is based on the multi-frontal solver MUMPS [1] and numerical results show that MUMPS can indeed handle matrices of this type reasonably well. Our MATLAB implementation also uses global matrices, with the MATLAB expressions involving sparse matrices operating on vectors in the space W just as in the formulas here.

However, if there are any constraints in the globs, one has to solve the constrained system (18), and MUMPS cannot do this directly. Thus, we will transform (18) to a symmetric, positive definite system that MUMPS can solve.

One way to solve system (18) is to introduce the orthogonal projection Π onto the nullspace of D^c, which is given by

$$\Pi = I - D^{c\mathrm{T}} \left(D^c D^{c\mathrm{T}}\right)^{-1} D^c. \tag{32}$$

Due to the block structure of D^c, where each block corresponds to a different glob and because by definition each degree of freedom belongs to at most one glob, the construction of Π can be performed in parallel.

Using projection Π, (18) is equivalent to

$$\Pi A^c \Pi w_c = \Pi R^{c\mathrm{T}} E^{\mathrm{T}} \left(I - P\right)^{\mathrm{T}} r, \quad w_c \in \operatorname{null} D^c. \tag{33}$$

However, the operator $\Pi A^c \Pi$ is singular for nontrivial D, so we solve instead a modified system

$$\left[\Pi A^c \Pi + t(I - \Pi)\right] w_c = \Pi R^{c\mathrm{T}} E^{\mathrm{T}} \left(I - P\right)^{\mathrm{T}} r, \tag{34}$$

where $t > 0$ is some scaling constant, e.g. chosen as the maximal diagonal entry in A^c. Now, the operator $\Pi A^c \Pi + t(I - \Pi)$ is regular, while the solutions of the systems (18) and (34) are the same.

The projection Π enforces constraints that couple all degrees of freedom on the same globs. For this reason, action of Π introduces new off-diagonal elements (called fill-in) in the projected system operator $\Pi A^c \Pi + t(I - \Pi)$. This is illustrated in Figure 3. Since the averages couple all variables in a glob, there are new dense off-diagonal blocks between globs, and the performance of sparse direct solvers would seriously deteriorate.

5 Generalized change of variables

To reduce the fill-in in enforcing the constraints following (32)–(33), we revisit and generalize the change of variables proposed in [11, 20]. On each substructure i, consider first the change of variables by the transformation

$$w_i^{\mathrm{new}} = H_i w_i, \quad H_i = \begin{bmatrix} H_i^{\mathrm{avg}} \\ 0 \quad I \end{bmatrix} = \begin{bmatrix} \bar{U} & \bar{V} \\ 0 & I \end{bmatrix}. \tag{35}$$

That is, the averages (the coarse degree of freedom) are at the beginning of the vector w_i^{new}, replacing the variables in w_i, and the remaining variables in w_i are unchanged. We assume that the vectors of weights in the averages are linearly independent, that is, H_i^{avg} has full row rank. While this assumption guarantees that there exists a square submatrix of H_i^{avg} consisting of linearly independent columns, this does not necessarily needs to be the matrix $\bar{U}$, so the inverse transformation H_i^{-1} may not exists. To correct this, we use QR decomposition with column pivoting to choose which variables in w will be replaced by the averages. Decompose

$$H_i^{\mathrm{avg}} = Q \begin{bmatrix} U & V \end{bmatrix} K,$$

where Q is orthogonal matrix, U is upper triangular square matrix, and K is a permutation matrix. We now define the generalized change of variables by

$$w_i^{\mathrm{new}} = H_i w_i, \quad H_i = \begin{bmatrix} U & V \\ 0 & I \end{bmatrix} K. \tag{36}$$

Now the inverse change of variable exists,

$$T_i = H_i^{-1} = K^{-1} \begin{bmatrix} U^{-1} & -U^{-1}V \\ 0 & I \end{bmatrix}. \tag{37}$$

The matrix U, though invertible, is not guaranteed to be well conditioned. This is a well-known problem in QR decomposition [17]. However, we can drop the rows of $[U\ V]$ where the diagonal entry of U is small, and one can argue that the constraints that were transformed into rows with negligible leading entry are (numerically) redundant. Our implementation of the change of variable uses QR decomposition by the LAPACK routine DGEQP3.

To compare with the change of variable from [11, 20], consider the case where there is just one average with unit weights. Then

$$H^{\text{avg}} = [1, \ldots, 1], \quad U = [1], \quad K = I, \quad H_i = \begin{bmatrix} 1 & 1 & \ldots & 1 \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{bmatrix},$$

and we have the change of variable

$$w_i = T_i w_i^{\text{new}}, \quad T_i = H_i^{-1} = \begin{bmatrix} 1 & -1 & \ldots & -1 \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{bmatrix}, \tag{38}$$

while the transformation of variable in [11, 20] is

$$w_i = \begin{bmatrix} 1 & -1 & \ldots & -1 \\ 1 & 1 & & \\ & & \ddots & \\ 1 & & & 1 \end{bmatrix} w_i^{\text{new}}.$$

With the change of basis, the BDDC preconditioner can be written as

$$M_{BDDC} : r \longmapsto u = (I - P)\, ETR\overline{w}, \quad \overline{w} \in \widetilde{W} : \quad \langle AT\overline{w}, Tz \rangle = \langle r, (I - P)\, ETz \rangle, \quad \forall z \in \widetilde{W},$$

where $T = \operatorname{diag}[T_i]$. Thus, A is replaced the transformed matrix $T^{\mathrm{T}}AT$, and, by assembly at corners following (13), A^c becomes $R^{c\mathrm{T}}T^{\mathrm{T}}ATR^c$. Then, Lemma 2 yields the matrix form of the algorithm: solving the system

$$\begin{array}{ccccc} R^{c\mathrm{T}}T^{\mathrm{T}}ATR^c\overline{w}_c & + & \overline{D}^{cT}\lambda & = & R^{c\mathrm{T}}T^{\mathrm{T}}E^{\mathrm{T}}r, \\ \overline{D}^c\overline{w}_c & & & = & 0, \end{array} \tag{39}$$

followed by computation of the approximate solution $u \in \widehat{W}$ by $u = (I - P)\, ETR^c\overline{w}_c$. Here, the matrix $\overline{D}^c = D^cT$ is much sparser than D^c because, thanks to the change of variables, it

couples only explicit new degrees of freedom on each subdomain and thus has only one $+1$ and one -1 entry on each row. In fact, the construction of $\overline{D}^c$ is similar to the construction of the operator B used in FETI methods. In computations, $\overline{D}^c$ can be constructed directly without using either D^c or T, knowing only which pairs of the interface degrees of freedom have to be coupled after the change of basis.

Instead of solving the saddle point problem (39) directly, we now use the projection as in (34) with D^c replaced by $\overline{D}^c$, resulting in a projection $\overline{\Pi}$. The sparsity structure of projection $\overline{\Pi}$ and the projected operator $\overline{\Pi}R^{c\mathrm{T}}T^{\mathrm{T}}ATR^c\overline{\Pi}+\bar{t}(I-\overline{\Pi})$ are illustrated Figure 3. As can be observed, change of basis preceding the projection can lead to much lower fill-in in the off-diagonal blocks of the projected matrix.

The BDDC preconditioner in an algebraic form can finally be rewritten in the form that is actually used in our implementation based on the MUMPS solver as follow.

Algorithm 2 *The action of the BDDC preconditioner $M_{BDDC} : r \mapsto w$ with the generalized change of variables consists of solving the system*

$$\widetilde{A}\overline{w}_c = \overline{\Pi}R^{c\mathrm{T}}T^{\mathrm{T}}E^{\mathrm{T}}r,$$

where

$$\widetilde{A} = \overline{\Pi}R^{c\mathrm{T}}T^{\mathrm{T}}ATR^c\overline{\Pi} + t(I-\overline{\Pi}), \tag{40}$$

with an arbitrary $t > 0$, followed by $w = (I-P)\,ETR^c\overline{w}_c$.

Remark 3 *Since the transformation of variables makes averages into separate degrees of freedom, one can treat these degrees of freedom as corners and assemble them just as in [11, 13, 20] to make all constraints primal. This gives no additional fill-in beyond the one caused by the change of variables, i.e., replacing A^c by $R^{c\mathrm{T}}T^{\mathrm{T}}ATR^c$. In the adaptive method (Section 3), the corners are already set and used to compute the constraints to be added adaptively. Treating all constraints as corners then requires redefining which variables are corners. This is not supported in the code described here. See Section 6 for more details.*

6 Implementation

We have first implemented the proposed method in MATLAB. Later, we have also developed a parallel version using Fortran 90 programming language with MPI.

First, we have implemented the BDDC preconditioner based on the formulation (18). In the case of corner constraints only, i.e. D is empty, the method is reduced to solving a problem with matrix A^c in each iteration. In the current version, we rely on the parallel direct solver MUMPS [1] (version 4.8.4) for this purpose.

In our implementation, two separate instances of MUMPS are necessary – one for solving problems with matrix A^c and another for a realization of the operator $I-P$ of the discrete harmonic extension in (8) globally, instead of solving an independent Dirichlet problem on each subdomain separately.

In the case of a nontrivial matrix D, i.e. for additional constraints on edges and/or faces, explicit change of variables with projection (Section 5) is performed in parallel to form the distributed sparse matrix (40), which is then supplied to MUMPS. We have observed a great advantage in projecting the matrix after the change of variables compared to the direct projection on null D. It greatly decreases time and memory consumption due to reduced fill-in, as described in Section 5. In our experience, the amount of extra work needed for the transformation and the projection is only a small fraction of the time saved by lower number of PCG iterations, compared to the case of corner constraints only.

The projection approach instead of assembling the matrix again after the change of variables allows us to store the sparse matrix in memory only once, and use it in the preconditioner as well as in the PCG method, which runs on vectors from the space $\widetilde{W}^c$, represented by their values on substructures' boundary. For the preconditioner, new entries arising from the transformation and projection are stored in the memory behind the original matrix and the convention of repeated indices allowed by MUMPS is exploited.

Later, the adaptive selection of constraints described in Section 3 has been added to the implementation. As the parallelization of solving generalized eigenvalue problems (28) on pairs of adjacent subdomains does not follow the scheme of the natural parallelization by subdomains, this part of the code has been written as a self-standing module that just passes the constraints to the main BDDC solver. Multiplication by S^c in the eigenvalue problem (28) is implemented by performing interior correction on each of the two adjacent subdomains separately, and only then assembling the result; thus, the matrices S^c and A^c for the two adjacent substructures are never formed explicitly.

7 Numerical results

We have tested the adaptive algorithm on several three-dimensional linear elasticity problems coming from engineering practice. As a consistency check, we have also tested the method in two dimensions with essentially the same results as in [27]. The computations were done in MATLAB and the parallel implementation described in Section 6. The generalized eigenvalue problems for pairs of adjacent substructures were solved by setting up the matrices and using standard methods for the symmetric eigenvalue problem in MATLAB, and we have also tested the MATLAB version of the LOBPCG algorithm [15]. The averaging operator was constructed with weights proportional to the diagonal entries of the substructure matrices before elimination of interiors.

The first problem is a nozzle box of a ŠKODA steam turbine 28 MW for the electric power plant Nováky, Slovakia, loaded by steam pressure. The body of the nozzle box was discretized using 2 696 isoparametric quadratic finite elements with 40 254 degrees of freedom and decomposed into 16 substructures with 37 corners, 19 edges and 32 faces, see Fig. 4. Convergence of the algorithm with non-adaptive constraints is displayed in Table 1. Note that the corner coarse degrees of freedom were not sufficient to guarantee the convergence. Comparing the last two rows in Table 1 we see that constraints obtained from the adaptive algorithm work quite better than arithmetic averages. Our explanation is that

such constraints might approximate better the direction of global eigenvectors corresponding to the extreme eigenvalues. Table 2 then contains results obtained using the adaptive selection of constraints. Each row corresponds to a different value of the threshold τ. All eigenvectors corresponding to eigenvalues greater or equal to τ were used to generate adaptive constraints. Comparing the results in Tables 1–2, we see that the adaptive method leads to a redistribution of the number of constraints on different faces and that, e.g., with $\tau = 20$ the total number of constraints is still lower compared to the total number of constraints obtained by using arithmetic averages, but the number of iterations is improved by almost 25% and the condition number estimate κ is improved by more than 50%.

The second problem is a beam with a mesh refinement around a notch, discretized using 245 687 tetrahedral finite elements with 143 451 degrees of freedom, and decomposed into 8 substructures with 31 corners, 18 edges and 19 faces, see Fig. 5. The results with non-adaptive constraints are summarized in Table 3, and results of the adaptive method in Table 4. Comparing these two tables, we see that, similarly as in the nozzle box problem, doubling the number of constraints reduces number of iterations to a half. Nevertheless, for both problems, the adaptive algorithms leads to a relatively small improvement in terms of number of iterations and condition number estimate. This indicates that for these problems the simple arithmetic averages already work well enough as there are no interfaces that would require extra work - the quality of the decomposition is uniform, as seen in Figures 4–5.

The power of the adaptive algorithm seems to dominate on finite element discretization with bad aspect ratios. An example of such problem is a bridge construction discretized by 39 060 hexahedral finite elements with 157 356 degrees of freedom, and distributed into 16 substructures with 250 corners, 30 edges and 43 faces, see Fig. 6. The results are summarized in Tables 5–6. Comparing the last two rows in Table 5, we see that relatively poor convergence with arithmetic averages improves quite significantly when arithmetic averages over faces are replaced by the same number of adaptive averages. Moreover, from Table 6 we see that, e.g., doubling the number of constraints with $\tau = 5$ decreases the number of iterations more than six times, and with $\tau = 2$ the number of iterations is reduced more than ten times while the number of constraints increases approximately four times.

In order to test the performance of the algorithm in the presence of jumps in material coefficients we have created a problem of a cube with material parameters $E = 10^6$ Pa and $\nu = 0.45$, penetrated by four bars with parameters $E = 2.1 \cdot 10^{11}$ Pa and $\nu = 0.3$, consisting of 107 811 degrees of freedom, and distributed into 8 substructures with 30 corners, 16 edges and 15 faces, see Fig. 7 (note that the bars cut the substructures only through faces). Similar problems are solved in practice to determine numerically (anisotropic) properties of composite materials. Comparing the results in Tables 7 and 8 we see that with $\tau = 10\,000$ only 10 additional averages over faces are used to decrease the number of iterations 2.6 times, and with $\tau = 2$ the number of iterations decreased 10 times compared to the non-adaptive algorithm with arithmetic averages over all globs (c+e+f) whereas the number of constraints increased less than three times.

To test the parallel behaviour of the MUMPS solver at our applications, we run the solver only with corner coarse degrees of freedom on a benchmark problem of cubic subdomains, the

number of which was growing in two dimensions (see Figure 8). As the size of the subdomains is fixed, the problem fixed at one side with load at the opposite side is changing its nature and so some growth in number of iterations is expected. The sequence of problems is run on increasing number of processors that matches the number of subdomains. Presented computations were performed on 1.5 GHz Intel Itanium 2 processors of SGI Altix 4700 computer in CTU Supercomputing Centre, Prague.

We can see in Table 9, that after a jump in times between 16 and 25 subdomains related probably to the computer's architecture, the times of analysis, factorization, as well as per one iteration remain almost constant. The 'total wall time' includes also the second factorization of MUMPS for computing the discrete harmonic extensions and all I/O operations.

In the presented algorithm, a considerable amount of work is spent in generating the adaptive constraints. This part, where a number of local eigenproblems is solved, may eventually dominate the whole computation. Each of these eigenproblems is common to two subdomains so the natural parallelism is different than the 'natural' one for domain decomposition methods, i.e. one processor per z subdomains, where z is an integer. For this reason, an independent distribution of eigenproblems among processors is performed and each processor which solves an eigenproblem is linked with the two processors which store the data of subdomains within the pair.

An investigation of scaling of this part on variable number of processors was performed on the problem of turbine nozzle box with 16 subdomains. A summary is shown in Figure 9 for the sequence of $2^k,\ k = 0, 1, 2, \ldots, 5$ processors.

8 Conclusion

The adaptive BDDC method has been presented. The paper contains several original contributions. First, the definition of space where BDDC runs is given as the nullspace of global matrix of constraints. For an efficient and straightforward implementation of this formulation, a generalization of change of variables is proposed. This allows efficient handling of multiple arbitrary constraints on a substructure face. This functionality is required for the implementation of constraints that are generated adaptively. The adaptive selection of constraints from [27] has been reformulated in a mathematically equivalent way to use only the operators of BDDC and to match the overall approach of the rest of the paper to minimize programming requirements. The adaptive method is based on simultaneous solution of generalized eigenvalue problems defined for each face in the decomposition. The eigenvalues serve as a condition number indicator, so the minimal number of constraints is added to guarantee that the condition number indicator is below a given threshold. The corresponding eigenvectors are used to derive the coefficients of the constraints. Numerical experiments confirm that the eigenvalues provide a good prediction of the final condition number of the preconditioned operator.

A parallel implementation of the method has been developed and presented. It is based on a global formulation of the matrix of the BDDC preconditioner, and it is built on top of solver MUMPS, which provides most of the parallelism and minimizes custom coding. The

implementation has been tested on a number of problems of 3D elasticity. Results for several real world problems are included.

In our experiments, adaptive BDDC has shown to be quite powerful. Many times, it has been able to save the situation in the case of poorly selected corners, even in the case of disconnected subdomains, a situation often faced in real applications of domain decomposition when using graph partitioners. Adaptive BDDC is able to handle very ill-conditioned problems (e.g. problems with jumps in coefficients, complicated geometries with deformed elements, etc.), which are almost impossible to solve by standard BDDC method using only arithmetic averages on edges and faces. Such problems would either require a prohibitive number of PCG iterations or may not converge at all. This class of problems is the target application of the proposed method, as the extra costs of generating the constraints adaptively is not negligible, and would not pay for well-conditioned problems.

The solution of local eigenproblems by LOBPCG to generate the adaptive constraints requires many iterations and accounts for most of the time for some problems. Preconditioning of the eigenproblems to reduce the number of iterations will be studied elsewhere.

Acknowledgement

We would like to thank to Jaroslav Kruis, Jaroslav Novotný and Jan Leština for providing us with real engineering problems. We are also grateful to Marian Brezina for visualization of some meshes by his own graphic tool. Part of this work was done when Jakub Šístek was visiting University of Colorado Denver.

This work was supported in part by National Science Foundation under grants DMS-0713876 and CNS-0719641, by Czech Science Foundation under grant GA ČR 106/08/0403, and by Academy of Sciences of the Czech Republic under grant AV0Z10190503.

References

[1] P. R. Amestoy, I. S. Duff, J.-Y. L'Excellent, Multifrontal parallel distributed symmetric and unsymmetric solvers, Comput. Methods Appl. Mech. Engrg. 184 (2000) 501–520.

[2] S. C. Brenner, L.-Y. Sung, BDDC and FETI-DP without matrices or vectors, Comput. Methods Appl. Mech. Engrg. 196 (8) (2007) 1429–1435.

[3] M. Brezina, C. Heberton, J. Mandel, P. Vaněk, An iterative method with convergence rate chosen a priori, UCD/CCM Report 140, University of Colorado at Denver, Denver, CO, `http://ccm.ucdenver.edu/reports/rep140.pdf` (1999).

[4] J.-M. Cros, A preconditioner for the Schur complement domain decomposition method, in: I. Herrera, D. E. Keyes, O. B. Widlund (eds.), Domain Decomposition

Methods in Science and Engineering, National Autonomous University of Mexico (UNAM), México, 2003, pp. 373–380, 14th International Conference on Domain Decomposition Methods, Cocoyoc, Mexico, January 6–12, 2002.

[5] J. W. Demmel, Applied numerical linear algebra, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1997.

[6] C. R. Dohrmann, A preconditioner for substructuring based on constrained energy minimization, SIAM J. Sci. Comput. 25 (1) (2003) 246–258.

[7] C. Farhat, M. Lesoinne, K. Pierson, A scalable dual-primal domain decomposition method, Numer. Linear Algebra Appl. 7 (2000) 687–714, Preconditioning techniques for large sparse matrix problems in industrial applications (Minneapolis, MN, 1999).

[8] C. Farhat, J. Mandel, The two-level FETI method for static and dynamic plate problems. I. An optimal iterative solver for biharmonic systems, Comput. Methods Appl. Mech. Engrg. 155 (1-2) (1998) 129–151.

[9] J. Fish, V. Belsky, Generalized aggregation multilevel solver, Internat. J. Numer. Methods Engrg. 40 (23) (1997) 4341–4361.

[10] Y. Fragakis, M. Papadrakakis, The mosaic of high performance domain decomposition methods for structural mechanics: Formulation, interrelation and numerical efficiency of primal and dual methods, Comput. Methods Appl. Mech. Engrg. 192 (2003) 3799–3830.

[11] A. Klawonn, O. Rheinbach, A parallel implementation of dual-primal FETI methods for three dimensional linear elasticity using a transformation of basis, SIAM J. Sci. Comput. 28 (5) (2006) 1886–1906.

[12] A. Klawonn, O. Rheinbach, O. B. Widlund, An analysis of a FETI-DP algorithm on irregular subdomains in the plane, SIAM J. Numer. Anal. 46 (5) (2008) 2484–2504.

[13] A. Klawonn, O. B. Widlund, Dual-primal FETI methods for linear elasticity, Comm. Pure Appl. Math. 59 (11) (2006) 1523–1572.

[14] A. Klawonn, O. B. Widlund, M. Dryja, Dual-primal FETI methods for three-dimensional elliptic problems with heterogeneous coefficients, SIAM J. Numer. Anal. 40 (1) (2002) 159–179.

[15] A. V. Knyazev, Toward the optimal preconditioned eigensolver: locally optimal block preconditioned conjugate gradient method, SIAM J. Sci. Comput. 23 (2) (2001) 517–541, copper Mountain Conference (2000).

[16] J. Kruis, Domain decomposition methods for distributed computing, Saxe-Coburg Publications, Kippen, Stirling, Scotland, 2006.

[17] J. Langou, Private communication (2008).

[18] P. Le Tallec, J. Mandel, M. Vidrascu, Balancing domain decomposition for plates, Contemp. Math. 180 (1994) 515–524, proceedings of the 7th International Symposium on Domain Decomposition Methods, Penn State, November 1993.

[19] P. Le Tallec, J. Mandel, M. Vidrascu, A Neumann-Neumann domain decomposition algorithm for solving plate and shell problems, SIAM J. Numer. Anal. 35 (1998) 836–867.

[20] J. Li, O. B. Widlund, FETI-DP, BDDC, and block Cholesky methods, Internat. J. Numer. Methods Engrg. 66 (2) (2006) 250–271.

[21] J. Mandel, Intelligent block iterative methods, in: J. Robinson (ed.), FEM Today and the Future, Robinson and Associates, Okehampton, Devon EX20 4NT, England, 1993, pp. 471–477, proceedings of the 7th World Congress on Finite Elements, Monte Carlo, November 1993.

[22] J. Mandel, An iterative solver for p-version finite elements in three dimensions, Comput. Methods Appl. Mech. Engrg. 116 (1-4) (1994) 175–183, iCOSAHOM'92 (Montpellier, 1992).

[23] J. Mandel, M. Brezina, Balancing domain decomposition for problems with large jumps in coefficients, Math. Comp. 65 (216) (1996) 1387–1401.

[24] J. Mandel, C. R. Dohrmann, Convergence of a balancing domain decomposition by constraints and energy minimization, Numer. Linear Algebra Appl. 10 (7) (2003) 639–659.

[25] J. Mandel, C. R. Dohrmann, R. Tezaur, An algebraic theory for primal and dual substructuring methods by constraints, Appl. Numer. Math. 54 (2) (2005) 167–193.

[26] J. Mandel, B. Sousedík, Adaptive coarse space selection in the BDDC and the FETI-DP iterative substructuring methods: Optimal face degrees of freedom, in: O. B. Widlund, D. E. Keyes (eds.), Domain Decomposition Methods in Science and Engineering XVI, Lecture Notes in Computational Science and Engineering, vol. 55, Springer-Verlag, 2006, pp. 421–428.

[27] J. Mandel, B. Sousedík, Adaptive selection of face coarse degrees of freedom in the BDDC and the FETI-DP iterative substructuring methods, Comput. Methods Appl. Mech. Engrg. 196 (8) (2007) 1389–1399.

[28] J. Mandel, B. Sousedík, BDDC and FETI-DP under minimalist assumptions, Computing 81 (2007) 269–280.

[29] J. Mandel, B. Sousedík, C. R. Dohrmann, Multispace and multilevel BDDC, Computing 83 (2-3) (2008) 55–85.

[30] J. Mandel, R. Tezaur, On the convergence of a dual-primal substructuring method, Numer. Math. 88 (2001) 543–558.

[31] J. Mandel, R. Tezaur, C. Farhat, A scalable substructuring method by Lagrange multipliers for plate bending problems, SIAM J. Numer. Anal. 36 (5) (1999) 1370–1391.

[32] C. Pechstein, R. Scheichl, Analysis of FETI methods for multiscale PDEs, Numer. Math. 111 (2) (2008) 293–333.

[33] C. Pechstein, R. Scheichl, Analysis of FETI methods for multiscale PDEs - Part II: Interface variations, Numerische Mathematik, submitted (2009).

[34] G. Poole, Y.-C. Liu, J. Mandel, Advancing analysis capabilities in ANSYS through solver technology, Electronic Transactions on Numerical Analysis 15 (2003) 106–121, tenth Copper Mountain Conference on Multigrid Methods, April 2001.

[35] J. Šístek, J. Novotný, J. Mandel, M. Čertíková, Burda, P., BDDC by a frontal solver and stress computation in a hip joint replacement, Mathematics and Computers in Simulation, in print, available online January 21, 2009, DOI `10.1016/j.matcom.2009.01.002` (2009).

[36] B. F. Smith, P. E. Bjørstad, W. D. Gropp, Domain decomposition : parallel multilevel methods for elliptic partial differential equations, Cambridge University Press, Cambridge, 1996.

[37] B. Sousedík, Comparison of some domain decomposition methods, Ph.D. thesis, Czech Technical University in Prague, Faculty of Civil Engineering, Department of Mathematics (2008).

[38] B. Sousedík, J. Mandel, On the equivalence of primal and dual substructuring preconditioners, Electron. Trans. Numer. Anal. 31 (2008) 384–402, published online September 30, 2009.
URL `http://etna.mcs.kent.edu/vol.31.2008/pp384-402.dir/pp384-402.html`

[39] A. Toselli, O. Widlund, Domain decomposition methods—algorithms and theory, vol. 34 of Springer Series in Computational Mathematics, Springer-Verlag, Berlin, 2005.

[40] O. Widlund, Accomodating irregular subdomains in domain decomposition theory, in: Eighteenth International Conference on Domain Decomposition, Springer-Verlag, 2008, to appear.

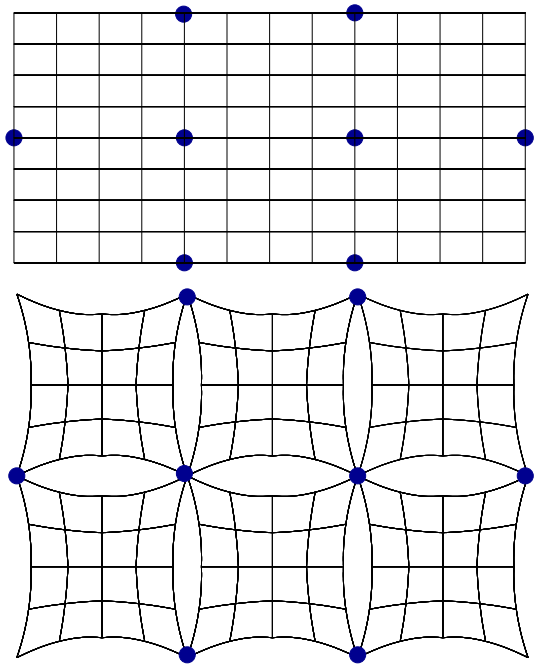


Figure 1: Example of an actual mesh (top) and the corresponding fictitious mesh for construction of space W^c (bottom), the dots mark corners.

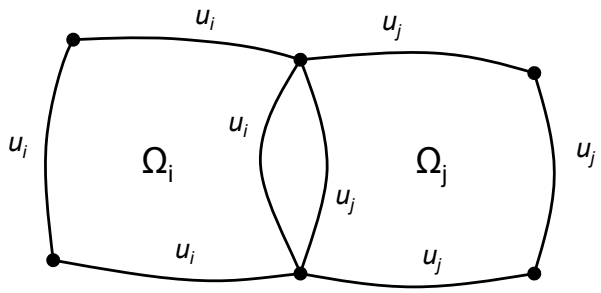
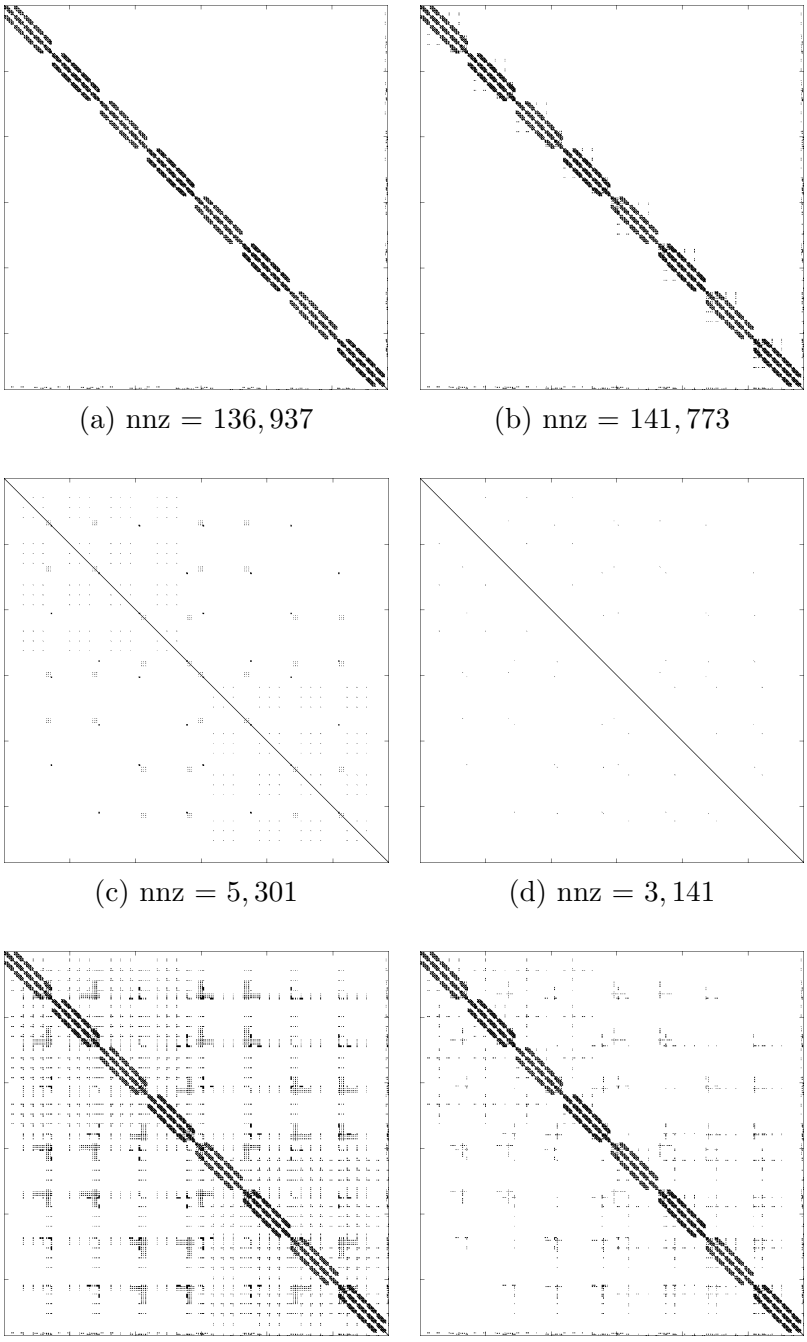


Figure 2: Illustration of two adjacent subdomains in 2D for the computation of the condition number indicator.



(a) nnz = 136,937 (b) nnz = 141,773

(c) nnz = 5,301 (d) nnz = 3,141

(e) nnz = 228,954 (f) nnz = 156,982

Figure 3: Sparsity patters for 3D elasticity problem for a cube decomposed into $2 \times 2 \times 2$ substructures ($H/h = 4$) with 7 corners, 6 edges, 12 faces, and 2,187 degrees of freedom. The matrix D^c (resp. $\overline{D}^c$) contains 54 rows to enforce the equality of arithmetic averages over edges. The matrices (a), (c), (e) are in the original degrees of freedom, while (b), (d), (f) are after the change of variables (37): the operators A^c in panel (a) and $R^{c\mathrm{T}} T^{\mathrm{T}} A T R^c$ in panel (b), projections Π in panel (c) and $\overline{\Pi}$ in panel (d), and projected operators $\Pi A^c \Pi + t(I - \Pi)$ in panel (e) and $\overline{\Pi} R^{c\mathrm{T}} T^{\mathrm{T}} A T R^c \overline{\Pi} + \bar{t}(I - \overline{\Pi})$ in panel (f). All are square matrices with size 2,925.

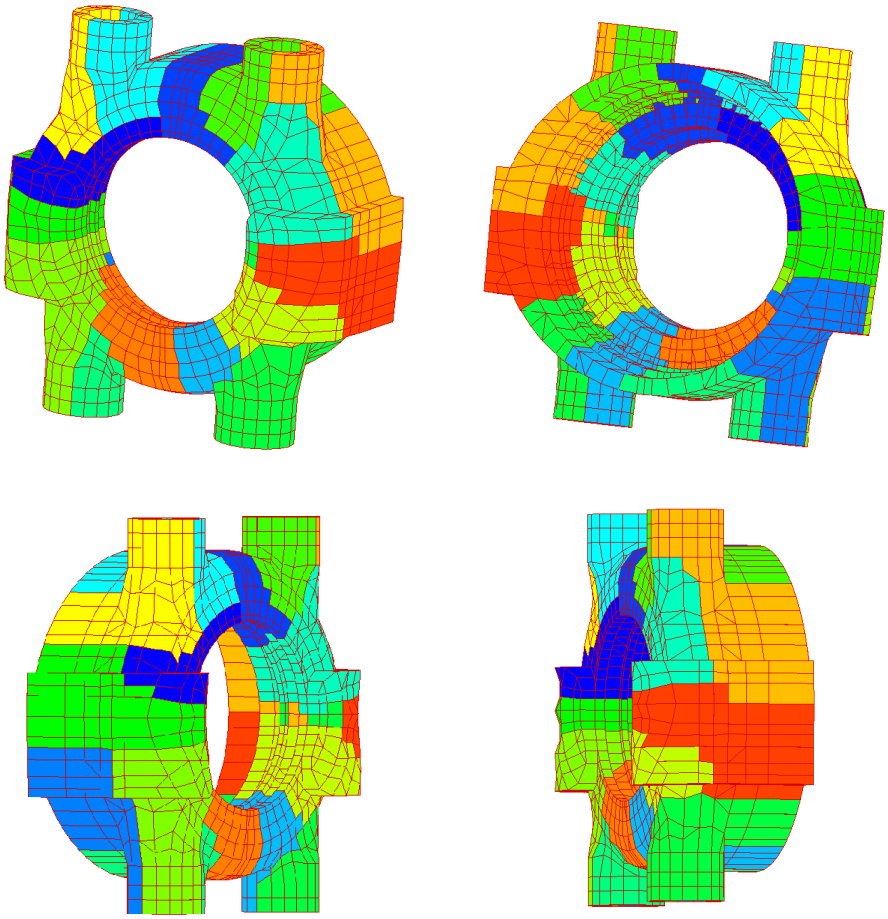


Figure 4: Finite element discretization and substructuring of the nozzle box, consisting of 40 254 degrees of freedom, 16 substructures, 37 corners, 19 edges and 32 faces.

constraints	Nc	κ	it
c	0	NA	NA
c+e	117	1 021.7	103
c+e+f	213	40.3	47
c+e+f (3eigv)	213	26.5	40

Table 1: Results for the turbine nozzle box problem. The first three rows correspond to non-adaptive approach with corner constraints and arithmetic averages over edges/faces, and the last row corresponds to corner constraints with arithmetic averages over edges and three weighted averages over faces obtained from eigenvectors of the local generalized eigenvalue problems, Nc is number of constraints (rows in the matrix D), κ is approximate condition number estimate from the Lanczos sequence in conjugate gradients, and it is number of iterations for relative residual tolerance 10^{-8}.

τ	$\widetilde{\omega}$	Nc	κ	it
∞(=c+e)	NA	117	1 021.7	103
50	49.8	158	44.9	48
20	19.8	200	16.9	36
10	> 10	274	11.2	27
5	> 5	408	8.8	20

Table 2: Results for the turbine nozzle box problem using the adaptive approach. τ is the threshold, and $\widetilde{\omega}$ is the condition number indicator from (26). The other headings are same as in Table 1.

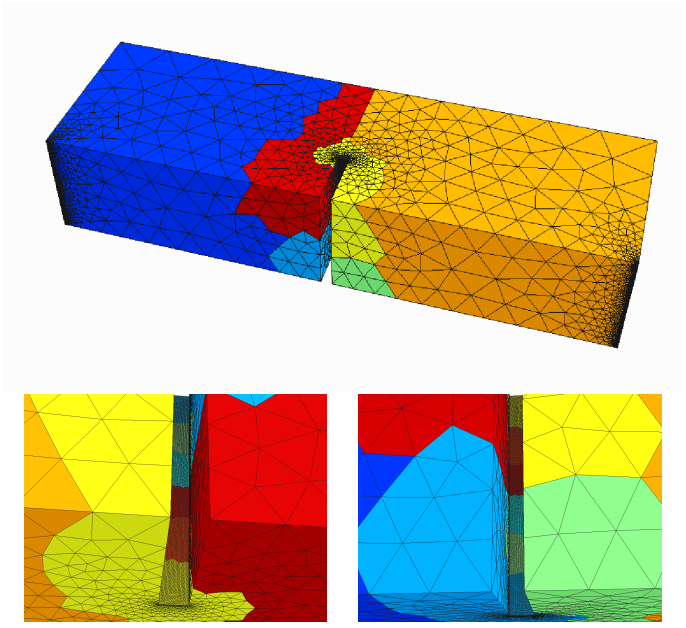


Figure 5: Finite element discretization and substructuring of the beam with a notch, consisting of 143 451 degrees of freedom, 8 substructures, 31 corners, 18 edges and 19 faces.

constraint	Nc	κ	it
c	0	127.1	79
c+e	111	101.0	61
c+e+f	168	22.4	32
c+e+f (3eigv)	168	13.2	30

Table 3: Results for the beam with a notch. The headings are same as in Table 1.

τ	$\widetilde{\omega}$	Nc	κ	it
∞(=c+e)	149.0	111	101.0	61
20	18.3	119	19.0	41
10	> 10	134	8.5	31
5	> 5	163	4.7	24
3	> 3	215	2.9	18
2	> 2	340	2.1	14

Table 4: Results for the beam with a notch. The headings are same as in Table 2.

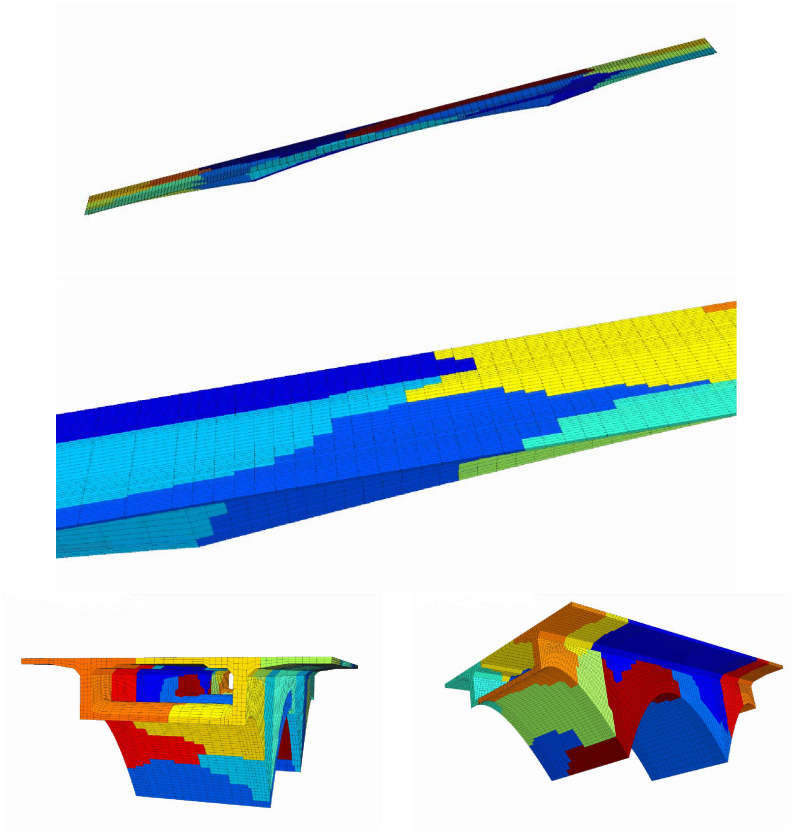


Figure 6: Finite element discretization and substructuring of the bridge construction, consisting of 157 356 degrees of freedom, 16 substructures, 250 corners, 30 edges and 43 faces.

constraint	Nc	κ	it
c	0	2 301.4	224
c+e	180	2 252.4	220
c+e+f	309	653.6	160
c+e+f (3eigv)	309	177.8	103

Table 5: Results for the bridge construction. The headings are same as in Table 1.

τ	$\widetilde{\omega}$	Nc	κ	it
∞(=c+e)	6 500.5	180	2 252.4	220
650	589.3	185	483.5	169
30	29.6	292	28.7	64
5	> 5	655	5.0	26
2	> 2	1301	2.0	14

Table 6: Results for the bridge construction. The headings are same as in Table 2.

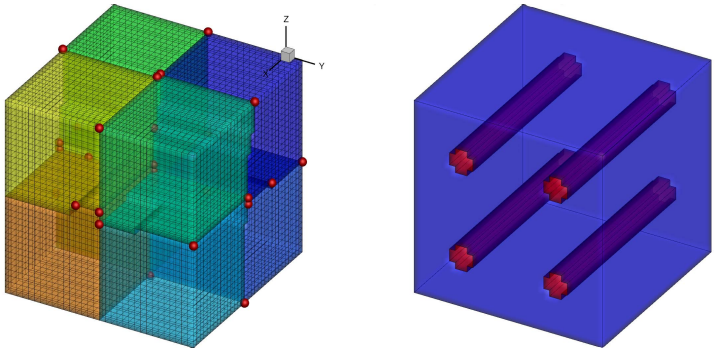


Figure 7: Finite element discretization and substructuring of the cube with jumps in coefficients, consisting of 107 811 degrees of freedom, 8 substructures, 30 corners, 16 edges and 15 faces.

constraint	Nc	κ	it
c	0	408 101.0	326
c+e	108	125 390.0	234
c+e+f	153	18 914.9	169
c+e+f (3eigv)	153	1 266.4	71

Table 7: Results for the cube with jumps in coefficients. The headings are same as in Table 1.

τ	$\widetilde{\omega}$	Nc	κ	it
∞(=c+e)	270 000.0	108	125 390.0	234
10 000	5 145.3	118	1 843.4	90
1 000	380.0	129	173.6	35
100	77.2	132	6.4	24
5	> 5	173	4.4	20
2	> 2	451	2.8	16

Table 8: Results for the cube with jumps in coefficients. The headings are same as in Table 2.

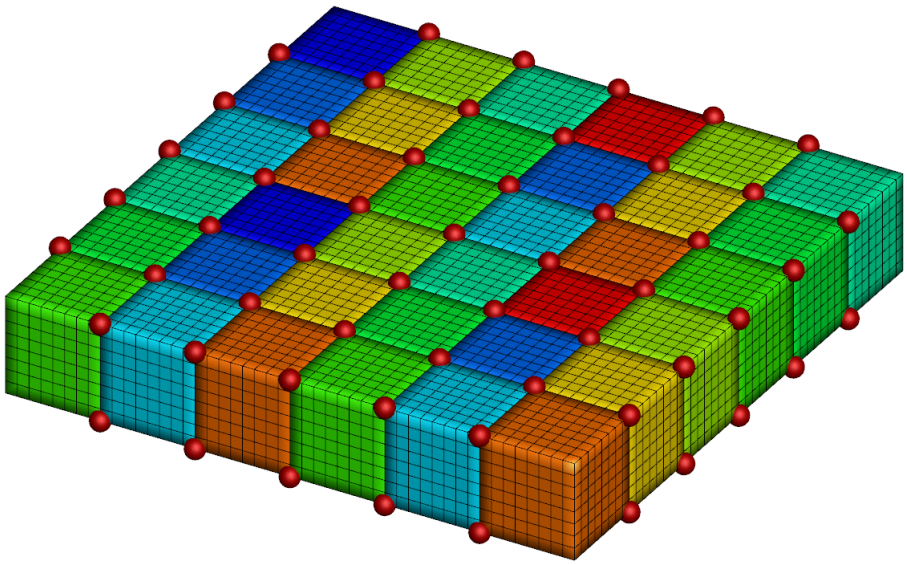


Figure 8: Example of a configuration of planar cubes test problem with 36 subdomains and H/h = 8, red dots represent corners.

number of subdomains	4	9	16	25	36	49	64
degrees of freedom	7 803	16 875	29 403	45 387	64 827	87 723	114 075
condition number est.	28.3	38.0	42.2	44.4	45.7	46.5	47.1
number of PCG iterations	13	26	36	42	44	46	47
analysis by MUMPS (sec)	0.2	0.5	1	15	14	16	19
factorization by MUMPS (sec)	0.5	0.4	0.8	12	10	12	14
PCG iterations (sec)	0.8	3.6	13	613	524	579	643
one PCG iteration (sec)	0.06	0.14	0.4	15	12	13	14
total wall time (sec)	3	6	19	715	616	696	794

Table 9: Weak scaling on planar cubes problem (e.g. Fig. 8), corners only , H/h = 8.

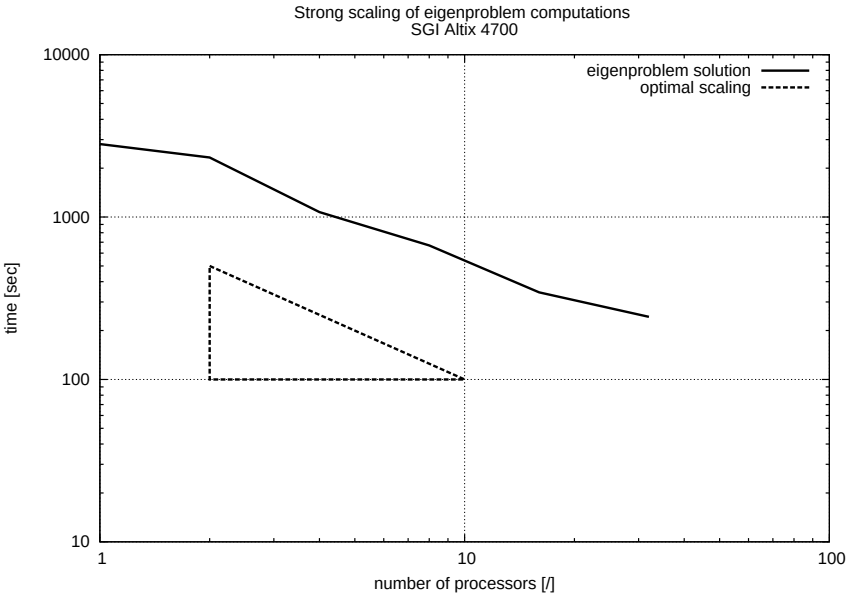


Figure 9: Dependence of computational time on number of processors for solution of local eigenproblems, nozzle box problem, 16 subdomains, 30 eigenproblems.