An Adaptive Multipreconditioned Conjugate Gradient Algorithm

Nicole Spillane

AN ADAPTIVE MULTIPRECONDITIONED CONJUGATE GRADIENT ALGORITHM

NICOLE SPILLANE [*]

Abstract. This article introduces and analyzes a new adaptive algorithm for solving symmetric positive definite linear systems in cases where several preconditioners are available or the usual preconditioner is a sum of contributions. A new theoretical result allows to select, at each iteration, whether a classical preconditioned CG iteration is sufficient (*i.e.*, the error decreases by a factor of at least some chosen ratio) or whether convergence needs to be accelerated by performing an iteration of multipreconditioned CG [4]. This is first presented in an abstract framework with the one strong assumption being that a bound for the smallest eigenvalue of the preconditioned operator is available. Then, the algorithm is applied to the Balancing Domain Decomposition method and its behaviour is illustrated numerically. In particular, it is observed to be optimal in terms of local solves, both for well-conditioned and ill-conditioned test cases, which makes it a good candidate to be a default parallel linear solver.

Key words. Krylov subspace methods, Preconditioners, Conjugate Gradient, Domain Decomposition, Robustness, BDD

AMS subject classifications. 65F10, 65N30, 65N55

1. Introduction. The problem of solving a symmetric positive definite linear system $\mathbf{A}\mathbf{x}_* = \mathbf{b}$ with the Conjugate Gradient (CG) algorithm [22] is considered in this article. Since these systems are possibly ill-conditioned, a very standard way to accelerate convergence is to use a preconditioner $\mathbf{H}$. This is to say that $\mathbf{H}\mathbf{A}\mathbf{x}_* = \mathbf{H}\mathbf{b}$ is solved where the condition number $\lambda_{\max}/\lambda_{\min}$ of $\mathbf{H}\mathbf{A}$ is much smaller than the one of $\mathbf{A}$ (with $\lambda_{\max}$ and $\lambda_{\min}$ denoting, respectively, the largest and smallest eigenvalue). The reason why this improves the convergence of the iterative solver is that the relative error at a given iteration depends on the condition number [34, 24, 42].

One particular type of preconditioning is projection preconditioning [9]. It is closely related to deflation [36, 43], augmentation [5] and balancing [23]. A review and analysis of these methods can be found in [48, 14] or [28] with an application to substructuring Domain Decomposition. The idea is to choose an auxiliary, or *deflation* space, and precompute the exact solution in this subspace. Then the iterative solver only needs to be applied to the remainder of the error. If the deflation space is well chosen the conditioning, and hence the convergence, are greatly improved. In practice, the deflation space is often computed as an approximation of the eigenvectors corresponding to isolated eigenvalues of the preconditioned operator, for example, by recycling information from a previous linear solve [15, 50, 2, 20] (see also section 2.2).

The method that is proposed is based on the MultiPreconditioned CG (MPCG) algorithm [4]. MPCG is itself related to the block CG algorithm [38, 37] and the multiparameter descent method [3]. It applies to linear systems for which there are N different preconditioners $\mathbf{H}^s$ ($s = 1, \dots, N$) or for which the usual preconditioner is a sum $\sum_{s=1}^{N} \mathbf{H}^s$. One case where this occurs is domain decomposition. Indeed the idea behind domain decomposition is to approximate the inverse of a global matrix by a sum of inverses of smaller *local* problems. This property has already been exploited since it has been proposed to compute the deflation space by solving local generalized eigenvalue problems: see [33] followed by [44, 47, 26, 27] for the substructuring

methods FETI and BDD [12, 29, 31]; [35, 8, 46, 13, 11] for the overlapping Additive Schwarz method [49]; and [30, 19] for the Optimized Schwarz method. In this article the Balancing Domain Decomposition method is used as an illustration for the proposed algorithms. Refer to section 4.3 for a brief summary of the deflation space proposed in [47] (called GenEO for Generalized Eigenvalues in the Overlaps) and the corresponding convergence result: fast convergence is guaranteed theoretically, even in hard heterogeneous cases such as the ones exhibited in [39].

The original motivation for the algorithms in this article was to compute the same deflation space without needing to solve generalized eigenvalue problems. As already mentioned, the framework for the proposed algorithms is MPCG [4]. MPCG has already been applied to Additive Schwarz [17] and FETI (Simultaneous FETI algorithm in [40, 16]). In both cases good convergence was observed. The drawback of these methods is that they generate a large number of search directions and the cost of minimizing over these search directions and orthogonalizing future contributions against them may become prohibitively expensive, especially if very many processors are used.

Instead, the newly proposed algorithms consider each contribution arising from the application of one $\mathbf{H}^s$ as a *candidate* [2] to augment the space in which the solution is sought for (called the minimization space). A theoretical estimate (2.4) predicts whether this candidate should augment the minimization space or whether it should only contribute to it through the global preconditioner $\sum_{s=1}^{N} \mathbf{H}^s$. The estimate is only practical if a lower bound for the eigenvalues of the preconditioned operator is known (*e.g.*, for the substructuring Domain Decomposition methods $\lambda_{\min} \geq 1$ [31, 29, 49]). The idea for the new adaptive MPCG algorithms was first briefly introduced in [45, section 7.3] in the FETI framework. It is proven that at each iteration, given a targeted contraction factor $0 < \rho < 1$, either the error is reduced by a factor ρ, or the coarse space is augmented with contributions coming from several components $\mathbf{H}^s$ (with the purpose of accelerating convergence). This guarantees that the iterations for which the algorithm performs some extra work are exactly those at which it is necessary.

The outline of the article is as follows. In section 2, some classical results for the Projected Preconditioned CG algorithm (PPCG) are introduced and the new estimate (2.4) is proven. In section 3, the two new algorithms are introduced in a general framework and the corresponding theoretical results are proven. In section 4, the algorithms are applied to BDD. Finally, in section 5, numerical experiments are presented which illustrate the behaviour of the proposed algorithms compared against other existing methods.

2. Projected Preconditioned Conjugate Gradient (PPCG). Let $n \in \mathbb{N}$. The three assumptions in this section are:

  (A1) $\mathbf{A} \in \mathbb{R}^{n \times n}$ is a symmetric positive definite matrix,

  (A2) $\mathbf{H} \in \mathbb{R}^{n \times n}$ is a symmetric positive definite matrix,

  (A3) $\mathbf{U} \in \mathbb{R}^{n \times n_0}$ is a full rank matrix with $n_0 < n$.

Throughout the article, the following problem is considered:

$$\text{Find } \mathbf{x}_* \in \mathbb{R}^n \text{ such that } \mathbf{A}\mathbf{x}_* = \mathbf{b}$$

for a given right hand side $\mathbf{b} \in \mathbb{R}^n$. The natural iterative solver for this problem is the conjugate gradient (CG) algorithm. Acceleration is performed by a (left) preconditioner denoted by $\mathbf{H}$, as well as a (right) projection preconditioner $\mathbf{\Pi}$ induced by the choice of $\mathbf{U}$ as follows. Let $\mathbf{\Pi}$ be the $\mathbf{A}$-orthogonal projection satisfying

$\text{Ker}(\mathbf{\Pi}) = \text{range}(\mathbf{U})$, or explicitly,

$$(2.1) \qquad \mathbf{\Pi} := \mathbf{I} - \mathbf{U}(\mathbf{U}^\top \mathbf{A} \mathbf{U})^{-1} \mathbf{U}^\top \mathbf{A}.$$

2.1. Description of PPCG and well known results. Algorithm 1 describes the Projected Preconditioned Conjugate Gradient (PPCG) algorithm [9] and introduces most of the notation.

Algorithm 1: PPCG algorithm for $\mathbf{A}\mathbf{x}_* = \mathbf{b}$ preconditioned by $\mathbf{H}$ and $\mathbf{\Pi} = \mathbf{I} - \mathbf{U}(\mathbf{U}^\top \mathbf{A} \mathbf{U})^{-1} \mathbf{U}^\top \mathbf{A}$ for initial guess $\mathbf{x}_{00}$

$\mathbf{x}_0 = \mathbf{\Pi} \mathbf{x}_{00} + \mathbf{U}(\mathbf{U}^\top \mathbf{A} \mathbf{U})^{-1} \mathbf{U}^\top \mathbf{b};$ `// Improved initial guess`

$\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$;

$\mathbf{z}_0 = \mathbf{H}\mathbf{r}_0;$

$\mathbf{p}_0 = \mathbf{\Pi}\mathbf{z}_0;$ `// Projected initial search direction`

for $i = 0, 1, \ldots,$ *convergence* **do**

> $\mathbf{q}_i = \mathbf{A}\mathbf{p}_i;$
>
> $\alpha_i = \dfrac{\langle \mathbf{r}_i, \mathbf{z}_i \rangle}{\langle \mathbf{q}_i, \mathbf{p}_i \rangle};$
>
> $\mathbf{x}_{i+1} = \mathbf{x}_i + \alpha_i \mathbf{p}_i$;
>
> $\mathbf{r}_{i+1} = \mathbf{r}_i - \alpha_i \mathbf{q}_i;$
>
> $\mathbf{z}_{i+1} = \mathbf{H}\mathbf{r}_{i+1};$
>
> $\beta_i = \dfrac{\langle \mathbf{z}_{i+1}, \mathbf{A}\mathbf{p}_i \rangle}{\langle \mathbf{p}_i, A\mathbf{p}_i \rangle};$
>
> $\mathbf{p}_{i+1} = \mathbf{\Pi}\mathbf{z}_{i+1} - \beta_i \mathbf{p}_i;$ `// Projected search direction`

end

Return $\mathbf{x}_{i+1};$

If $\mathbf{U}$ is the empty matrix then $\mathbf{\Pi} = \mathbf{I}$ and the usual Preconditioned Conjugate Gradient (PCG) algorithm is recovered. Algorithmically there are two differences between PCG and PPCG. The first is that the initial guess $\mathbf{x}_{00}$ given by the user is improved by computing the exact solution in the space range($\mathbf{U}$) (this can be rewritten as a projection): $\mathbf{x}_0 = \mathbf{\Pi}\mathbf{x}_{00} + \mathbf{U}(\mathbf{U}^\top \mathbf{A} \mathbf{U})^{-1} \mathbf{U}^\top \mathbf{b} = \mathbf{x}_{00} + (\mathbf{I} - \mathbf{\Pi})(\mathbf{x}_* - \mathbf{x}_{00})$. The second difference is a consequence. Indeed, the iterative solver must only compute the remaining part of the solution ($\mathbf{\Pi}\mathbf{x}_*$) so all search directions are projected into range($\mathbf{\Pi}$). Variants of this process are *deflation*, *balancing*, and *augmentation* [9, 36, 43, 5, 23, 14] (see, *e.g.*, [48] for a comparison). In domain decomposition the deflation space range($\mathbf{U}$) is referred to as the coarse space (see section 3). Next, a list of well known results for PPCG [28, 42] is stated:

1. The exact solution is achieved after at most $n - n_0 \ (= n - \text{rank}(\mathbf{U}))$ iterations.
2. $\|\mathbf{x}_* - \mathbf{x}_i\|_{\mathbf{A}} = \min \{ \|\mathbf{x}_* - \mathbf{x}\|_{\mathbf{A}}; \ \mathbf{x} \in \mathbf{x}_{00} + \text{range}(\mathbf{U}) + \text{span}\{\mathbf{p}_0, \ldots, \mathbf{p}_{i-1}\} \}$.
3. Search directions are pairwise $\mathbf{A}$-orthogonal: $\langle \mathbf{p}_i, \mathbf{A}\mathbf{p}_j \rangle = 0 \ (i \neq j)$.
4. Residuals are pairwise $\mathbf{H}$-orthogonal: $\langle \mathbf{r}_i, \mathbf{H}\mathbf{r}_j \rangle = 0$ for all $(i \neq j)$.

2.2. Choice of the deflation space. Since the approximation given by iteration i of PPCG minimizes the error over all vectors $\mathbf{x}_i \in \mathbf{x}_{00} + \text{range}(\mathbf{U}) + \text{span}\{\mathbf{p}_0, \ldots, \mathbf{p}_{i-1}\}$ it is natural that augmenting the space range($\mathbf{U}$) leads to better convergence. On the other hand, if the number n_0 of columns in $\mathbf{U}$ is too large, then the factorization of $\mathbf{U}^\top \mathbf{A} \mathbf{U}$ in the definition of $\mathbf{\Pi}$ becomes excessively costly. In other words, it is necessary to identify carefully which vectors help accelerate convergence.

One way to estimate the relative error of PCG is to use the following convergence result [34, 24] (see also [42, Theorem 6.29]):

$$(2.2) \qquad \frac{\|\mathbf{x}_* - \mathbf{x}_i\|_{\mathbf{A}}}{\|\mathbf{x}_* - \mathbf{x}_0\|_{\mathbf{A}}} \leq 2 \left[\frac{\sqrt{\lambda_{\max}/\lambda_{\min}} - 1}{\sqrt{\lambda_{\max}/\lambda_{\min}} + 1} \right]^i,$$

where $\lambda_{\max}$ and $\lambda_{\min}$ are bounds for the spectrum of the preconditioned operator $\mathbf{HA}$. For PPCG the same estimate holds but where $\lambda_{\max}$ and $\lambda_{\min}$ are replaced by bounds for the non zero eigenvalues of the projected preconditioned operator $\mathbf{HA\Pi}$ [9] (or equivalently the eigenvalues of $\mathbf{HA}$ restricted to range($\mathbf{\Pi}$)). Consequently, the ideal strategy for choosing $\mathbf{U}$ is to first compute all isolated eigenvalues of $\mathbf{HA}$ and use the corresponding eigenvectors as a basis for the deflation space range($\mathbf{U}$). This way, the spectrum of $\mathbf{HA\Pi}$ is clustered and (2.2) guarantees good convergence. Since it is unrealistic to compute the spectrum of $\mathbf{HA}$, there are proposals for approximating *a priori* the isolated eigenvalues. One option, popular in domain decomposition, is to solve auxiliary (less costly) eigenvalue problems [33, 44, 47, 26, 27, 35, 8, 46, 13, 11, 30, 19] (see also section 4.3).

The algorithms that are proposed in this article are closely related to deflation except that $\mathbf{U}$ is initialized with vectors chosen *a priori* and then augmented with vectors selected on the fly. First, an estimate that allows to do that is derived.

2.3. Monitoring the relative error in PPCG. With the notation from Algorithm 1, and denoting by $\mathbf{d}_i$ the error at iteration i: $\mathbf{d}_i = \mathbf{x}_* - \mathbf{x}_i$, the authors in [1] prove that

$$(2.3) \qquad \|\mathbf{d}_i\|_{\mathbf{A}}^2 = \|\mathbf{d}_{i-1}\|_{\mathbf{A}}^2 - \alpha_{i-1}^2 \|\mathbf{p}_{i-1}\|_{\mathbf{A}}^2, \text{ for all } i = 1, \ldots, n - n_0 - 1.$$

The proof consists of three steps: first, by the finite termination property (item 1 in section 2.1), the exact solution can be written as $\mathbf{x}_* = \mathbf{x}_0 + \sum_{j=0}^{n-n_0-1} \alpha_j \mathbf{p}_j = \mathbf{x}_i + \sum_{j=i}^{n-n_0-1} \alpha_j \mathbf{p}_j$. Then, the $\mathbf{A}$-conjugacy between search directions (item 3 in section 2.1) implies that $\|\mathbf{d}_i\|_{\mathbf{A}}^2 = \|\mathbf{x}_* - \mathbf{x}_i\|_{\mathbf{A}}^2 = \sum_{j=i}^{n-n_0-1} \alpha_j^2 \|\mathbf{p}_j\|_{\mathbf{A}}^2$. Finally, (2.3) follows easily by subtraction.

REMARK 1. *The proof in [1] is for the non projected PCG ($n_0 = 0$) but goes through in the same way when considering PPCG with the difference that the process terminates in at most $n - n_0$ iterations (instead of n).*

The authors use this to derive some *a posteriori* error estimates and stopping criteria. Here, from the same starting point two adaptive algorithms are derived with the objective of accelerating convergence when necessary. Assume that at iteration i the exact solution has not yet been found (*i.e.*, $\mathbf{d}_i \neq \mathbf{0}$) then (2.3) can be rewritten as

$$\frac{\|\mathbf{d}_{i-1}\|_{\mathbf{A}}^2}{\|\mathbf{d}_i\|_{\mathbf{A}}^2} = 1 + \frac{\|\alpha_{i-1}\mathbf{p}_{i-1}\|_{\mathbf{A}}^2}{\|\mathbf{d}_i\|_{\mathbf{A}}^2} = 1 + \frac{\|\alpha_{i-1}\mathbf{p}_{i-1}\|_{\mathbf{A}}^2}{\|\mathbf{r}_i\|_{\mathbf{H}}^2} \frac{\|\mathbf{r}_i\|_{\mathbf{H}}^2}{\|\mathbf{d}_i\|_{\mathbf{A}}^2} = 1 + \frac{\|\alpha_{i-1}\mathbf{p}_{i-1}\|_{\mathbf{A}}^2}{\|\mathbf{r}_i\|_{\mathbf{H}}^2} \frac{\|\mathbf{d}_i\|_{\mathbf{AHA}}^2}{\|\mathbf{d}_i\|_{\mathbf{A}}^2}.$$

Noting that the last factor is related to a Rayleigh quotient for $\mathbf{HA}$ it follows that

$$(2.4) \qquad \frac{\|\mathbf{d}_i\|_{\mathbf{A}}^2}{\|\mathbf{d}_{i-1}\|_{\mathbf{A}}^2} \leq \left(1 + \lambda_{\min} \frac{\|\alpha_{i-1}\mathbf{p}_{i-1}\|_{\mathbf{A}}^2}{\|\mathbf{r}_i\|_{\mathbf{H}}^2} \right)^{-1},$$

where $\lambda_{\min}$ is again the smallest eigenvalue of the preconditioned operator $\mathbf{HA}$. Indeed, it holds that $\|\mathbf{y}\|_{\mathbf{AHA}}^2 \geq \lambda_{\min} \|\mathbf{y}\|_{\mathbf{A}}^2$ for all $\mathbf{y} \in \mathbb{R}^n$ (see, *e.g.*, [49, Lemma C.1]).

From estimate (2.4) it can be deduced that, if there exists $\tau > 0$ such that $\tau\|\mathbf{r}_i\|_{\mathbf{H}}^2 \leq \|\alpha_{i-1}\mathbf{p}_{i-1}\|_{\mathbf{A}}^2$ at every iteration $i = 1,\ldots,j$, then $\|\mathbf{d}_j\|_{\mathbf{A}}/\|\mathbf{d}_0\|_{\mathbf{A}} \leq (1 + \lambda_{\min}\tau)^{-j/2}$. Conversely, to guarantee that the error decreases at least linearly with a given contraction factor ρ (*i.e.*, $\|\mathbf{d}_i\|/\|\mathbf{d}_{i-1}\| \leq \rho$), it is sufficient to check that:

$$(2.5) \qquad \frac{\|\alpha_{i-1}\mathbf{p}_{i-1}\|_{\mathbf{A}}^2}{\|\mathbf{r}_i\|_{\mathbf{H}}^2} \geq \tau \text{ with } \tau := \frac{1-\rho^2}{\lambda_{\min}\rho^2}.$$

In the next section two new algorithms are introduced that aim at guarantying a targeted convergence bound. They are based on evaluating, at each iteration, whether (2.5) holds or not. In the case where it doesn't, a way to accelerate convergence is proposed.

REMARK 2. *After division by* $\|\mathbf{d}_{i-1}\|_{\mathbf{A}}^2$, *(2.3) can also be rewritten as*

$$\frac{\|\mathbf{d}_i\|_{\mathbf{A}}^2}{\|\mathbf{d}_{i-1}\|_{\mathbf{A}}^2} = 1 - \frac{\langle \mathbf{r}_{i-1}, \mathbf{z}_{i-1}\rangle^2}{\langle \mathbf{Ap}_{i-1}, \mathbf{p}_{i-1}\rangle^2} \cdot \frac{\langle \mathbf{Ap}_{i-1}, \mathbf{p}_{i-1}\rangle}{\langle \mathbf{Ad}_{i-1}, \mathbf{d}_{i-1}\rangle} \leq 1 - \lambda_{\min}\frac{\langle \mathbf{r}_{i-1}, \mathbf{z}_{i-1}\rangle}{\langle \mathbf{Ap}_{i-1}, \mathbf{p}_{i-1}\rangle}.$$

This estimate is mentioned because it arises more naturally and is new. However, it is not used in the proposed adaptive algorithms.

REMARK 3. *The very existence of formula* (2.4) *motivates the choice of an adaptive MPCG algorithm over an adaptive MPGMRES (multipreconditioned GMRES) algorithm based on MPGMRES [18].*

3. Main Result: Adaptive MPCG. Two extra assumptions on the preconditioned system $\mathbf{HAx}_* = \mathbf{Hb}$ are made:

(A4) a lower bound $\lambda_{\min}$ for the spectrum of $\mathbf{HA}$ is known,

(A5) the preconditioner $\mathbf{H}$ is a sum of N contributions : $\mathbf{H} = \sum\limits_{s=1}^{N} \mathbf{H}^s$, with each $\mathbf{H}^s$ symmetric and positive semidefinite.

The motivation for these two assumptions is directly connected to the two main ingredients in the adaptive step of the proposed algorithm. Assumption **(A4)** guarantees that the terms in the relative error estimate (2.4) can be evaluated and as a consequence that this estimate can be used as an indicator of whether the algorithm needs to be adapted or not (*i.e.*, convergence needs to be accelerated). Assumption **(A5)** is just as vital since, when a lack of robustness is detected, convergence is improved by searching for the next approximate solution in a larger space spanned by contributions from each of the $\mathbf{H}^s$ instead of just one contribution corresponding to $\mathbf{H}$.

3.1. Presentation of the New Algorithm. Algorithm 2 presents the new adaptive MPCG algorithm and introduces some new notation. Adaptive MPCG is designed to adapt automatically if convergence is too slow. More precisely, given a threshold $\tau \in \mathbb{R}^+$ chosen by the user, the adaptive steps are between lines 7 and 12. The test in line 8 is referred to as the τ–test. If the τ–test returns $t_i \geq \tau$, then the algorithm predicts that there is no need to adapt and performs a step of PPCG. If the τ–test returns $t_i < \tau$, then the PPCG algorithm is not reducing the error sufficiently and the algorithm performs one step of the (projected) MPCG algorithm [4] for the N preconditioners $\mathbf{H}^s$. This automatically improves the next approximate solution since the error is minimized over a larger search space that includes the usual search space (see Property 2 in Theorem 3.1). There are two extreme choices for τ: if $\tau = 0$ then the usual PPCG iterations are recovered and if $\tau = \infty$ then the (projected) MPCG algorithm is recovered.

There is some connection to the selective MPGMRES algorithm proposed in [18]: with MPGMRES the growth of the minimization space is exponential and to reduce it to a linear growth the authors propose to select one search direction per preconditioner per iteration. Note however that with MPCG the growth is already linear and adaptive MPCG reduces that even further. Moreover, the τ-test provides a way to only add extra search directions when convergence is slow. This adaptive process is what makes Adaptive MPCG very efficient.

Algorithm 2: Adaptive multipreconditioned CG for $\mathbf{A}\mathbf{x}_* = \mathbf{b}$ preconditioned by $\left(\sum_{s=1}^{N} \mathbf{H}^s \right)$ and $\mathbf{\Pi}$ for initial guess $\mathbf{x}_{00}$. $\tau \in \mathbb{R}^+$: chosen by user.

---

1   $\mathbf{x}_0 = \mathbf{\Pi}\mathbf{x}_{00} + \mathbf{U}(\mathbf{U}^\top\mathbf{A}\mathbf{U})^{-1}\mathbf{U}^\top\mathbf{b}$; $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$; $\mathbf{Z}_0 = \mathbf{H}\mathbf{r}_0$; $\mathbf{P}_0 = \mathbf{\Pi}\mathbf{Z}_0$;

2   **for** $i = 0, 1, \ldots,$ *convergence* **do**

3      $\mathbf{Q}_i = \mathbf{A}\mathbf{P}_i$;

4      $\mathbf{\Delta}_i = \mathbf{Q}_i^\top\mathbf{P}_i$; $\boldsymbol{\gamma}_i = \mathbf{P}_i^\top\mathbf{r}_i$; $\boldsymbol{\alpha}_i = \mathbf{\Delta}_i^\dagger\boldsymbol{\gamma}_i$;

5      $\mathbf{x}_{i+1} = \mathbf{x}_i + \mathbf{P}_i\boldsymbol{\alpha}_i$;

6      $\mathbf{r}_{i+1} = \mathbf{r}_i - \mathbf{Q}_i\boldsymbol{\alpha}_i$;   *Adaptive Multipreconditioning*

7      $t_i = \dfrac{\boldsymbol{\gamma}_i^\top\boldsymbol{\alpha}_i}{\mathbf{r}_{i+1}^\top\mathbf{H}\mathbf{r}_{i+1}}$;

8   **if** $t_i < \tau$ **then**   // τ-test

9         $\mathbf{Z}_{i+1} = \left[\mathbf{H}^1\mathbf{r}_{i+1} \,\middle|\, \ldots \,\middle|\, \mathbf{H}^N\mathbf{r}_{i+1} \right]$;   // Concatenate the N vectors

10   **else**

11        $\mathbf{Z}_{i+1} = \mathbf{H}\mathbf{r}_{i+1}$;

12   **end**

13      $\mathbf{\Phi}_{i,j} = \mathbf{Q}_j^\top\mathbf{Z}_{i+1}$; $\boldsymbol{\beta}_{i,j} = \mathbf{\Delta}_j^\dagger\mathbf{\Phi}_{i,j}$ for each $j = 0, \ldots, i$;

14      $\mathbf{P}_{i+1} = \mathbf{\Pi}\mathbf{Z}_{i+1} - \sum_{j=0}^{i} \mathbf{P}_j\boldsymbol{\beta}_{i,j}$;

15   **end**

16   Return $\mathbf{x}_{i+1}$;

Each time $t_i < \tau$, an $N \times N$ matrix $\mathbf{\Delta}_i = \mathbf{P}_i^T\mathbf{A}\mathbf{P}_i$ must be inverted. Since $\mathbf{P}_i$ is the concatenation of contributions from the components $\mathbf{H}^s$ in the preconditioner, it is reasonable to expect that $\mathbf{\Delta}_i$ be full-rank. If it is not, then $\mathbf{\Delta}_i$ is only positive semidefinite and pseudo-inversion (denoted by $\mathbf{\Delta}_i^\dagger$) is necessary. In any case, both $\boldsymbol{\gamma}_i$ and the columns in $\mathbf{\Phi}_{j,i}$ are in $\text{range}(\mathbf{P}_i^T) = \text{range}(\mathbf{\Delta}_i)$ so that the iteration is always well defined. The exact same occurs in the Simultaneous FETI and Block FETI algorithms [16]. There, it is proposed to operate a rank-revealing Cholesky factorization (symmetric pivoting) on each $\mathbf{\Delta}_i$ to replace $\mathbf{P}_i$ by an $\mathbf{A}$-orthonormal basis $\overline{\mathbf{P}_i}$ of $\text{range}(\mathbf{P}_i)$ and simplify future orthogonalization steps.

The presented algorithm is with full reorthogonalization (lines 13 and 14). It is known that for MPCG this is necessary but for PPCG it is not ($\boldsymbol{\beta}_{i,j} = 0$ as soon as $j \neq i$) so some reorthogonalization steps may be skipped. Here, this is not further considered for several reasons: *(i)* adaptive MPCG is expected to solve the problem in few iterations so the cost of reorthogonalization will be low, *(ii)* for substructuring methods, which is the application considered in the next section, it has been observed that full reorthogonalization is in fact crucial because of numerical errors.

REMARK 4. *For clarity the size of the different variables is given (recall that n*

is the size of the global problem and the preconditioner has N components):

- $\mathbf{A}, \mathbf{\Pi}, \mathbf{H}, \mathbf{H}^0, \ldots, \mathbf{H}^N \in \mathbb{R}^{n \times n}$,
- $\mathbf{x}_*, \mathbf{x}_i, \mathbf{r}_i, \mathbf{b} \in \mathbb{R}^n$,
- $\mathbf{Z}_{i+1}, \mathbf{P}_{i+1}, \mathbf{Q}_{i+1} \in \mathbb{R}^{n \times N}$ *or* $\mathbb{R}^n$ *depending on the iteration,*
- $\mathbf{\Delta}_i \in \mathbb{R}^{N \times N}$ *or* $\mathbb{R}$ *depending on the iteration,*
- $\boldsymbol{\gamma}_i, \boldsymbol{\alpha}_i \in \mathbb{R}^N$ *or* $\mathbb{R}$ *depending on the iteration number i,*
- $\mathbf{\Phi}_{i,j}, \boldsymbol{\beta}_{i,j} \in \mathbb{R}^{N \times N}$ *or* $\mathbb{R}^N$ *or* $\mathbb{R}^{1 \times N}$ *or* $\mathbb{R}$ *depending on the iteration numbers i and j,*

Note that the quantities $\mathbf{P}_i \boldsymbol{\alpha}_i$ and $\mathbf{A}\mathbf{P}_i \boldsymbol{\alpha}_i$ are always vectors in $\mathbb{R}^n$. Note also that the notation $\langle \cdot, \cdot \rangle$ is only used for computing the inner product between two vectors.

3.2. The usual PPCG properties hold for Algorithm 2. In Theorem 3.1, results similar to the ones stated for PPCG in section 2.1 are proven for Algorithm 2.

THEOREM 3.1. *Algorithm 2 satisfies the five properties:*

1. *The exact solution is achieved after at most $n - n_0$ iterations.*
2. $\|\mathbf{x}_* - \mathbf{x}_i\|_{\mathbf{A}} = \min \left\{ \|\mathbf{x}_* - \mathbf{x}\|_{\mathbf{A}}; \mathbf{x} \in \mathbf{x}_{00} + range(\mathbf{U}) + \sum_{j=0}^{i-1} range(\mathbf{P}_j) \right\}$.
3. *Blocks of search directions are pairwise $\mathbf{A}$-orthogonal: $\mathbf{P}_j^\top \mathbf{A}\mathbf{P}_i = \mathbf{0}$ $(i \neq j)$.*
4. *Residuals are ℓ_2-orthogonal to previous search directions: $\mathbf{P}_j^\top \mathbf{r}_i = \mathbf{0}$ $(i > j)$.*
5. *Residuals are pairwise $\mathbf{H}$-orthogonal: $\langle \mathbf{H}\mathbf{r}_j, \mathbf{r}_i \rangle = 0$ $(i \neq j)$.*

Proof. In the following, many simplifications occur thanks to the $\mathbf{A}$-orthogonality of the projection matrix $\mathbf{\Pi}$. Also note that $\mathbf{\Pi}^\top \mathbf{A} = \mathbf{A}\mathbf{\Pi}$ by definition of $\mathbf{\Pi}$ in (2.1); $\mathbf{\Pi}\mathbf{P}_i = \mathbf{P}_i$ since $\mathbf{P}_i \in range(\mathbf{\Pi})$; and $\mathbf{\Pi}^\top \mathbf{r}_i = \mathbf{r}_i$ since $\mathbf{r}_i \in range(\mathbf{\Pi}^\top)$.

Proof by induction of Properties 3 and 4:

The case $i = 1$ is simple: $\mathbf{P}_0^\top \mathbf{A}\mathbf{P}_1 = \mathbf{P}_0^\top \mathbf{A}\mathbf{\Pi}\mathbf{Z}_1 - \mathbf{P}_0^\top \mathbf{A}\mathbf{P}_0 \boldsymbol{\beta}_{0,0} = \mathbf{\Phi}_{0,0} - \mathbf{\Delta}_0 \mathbf{\Delta}_0^\dagger \mathbf{\Phi}_{0,0} = \mathbf{0}$, and $\mathbf{P}_0^\top \mathbf{r}_1 = \mathbf{P}_0^\top \mathbf{r}_0 - \mathbf{P}_0^\top \mathbf{Q}_0 \boldsymbol{\alpha}_0 = \boldsymbol{\gamma}_0 - \mathbf{\Delta}_0 \mathbf{\Delta}_0^\dagger \boldsymbol{\gamma}_0 = \mathbf{0}$.

Next, assume that both properties hold for a given $i \geq 1$, the objective is to deduce them for $i + 1$. Let $j \leq i$, then

$$\mathbf{P}_j^\top \mathbf{A}\mathbf{P}_{i+1} = \mathbf{P}_j^\top \mathbf{A}\mathbf{\Pi}\mathbf{Z}_{i+1} - \sum_{k=0}^{i} \mathbf{P}_j^\top \mathbf{A}\mathbf{P}_k \boldsymbol{\beta}_{i,k} = \mathbf{\Phi}_{i,j} - \mathbf{\Delta}_j \mathbf{\Delta}_j^\dagger \mathbf{\Phi}_{i,j} = \mathbf{0},$$

$$\mathbf{P}_j^\top \mathbf{r}_{i+1} = \mathbf{P}_j^\top \mathbf{r}_i - \mathbf{P}_j^\top \mathbf{Q}_i \boldsymbol{\alpha}_i = \begin{cases} \mathbf{0} \text{ if } j \neq i \text{ since } \mathbf{P}_j^\top \mathbf{r}_i = \mathbf{0} \text{ and } \mathbf{P}_j^\top \mathbf{Q}_i = \mathbf{0}, \\ \mathbf{P}_i^\top \mathbf{r}_i - \mathbf{P}_i^\top \mathbf{Q}_i \boldsymbol{\alpha}_i = \boldsymbol{\gamma}_i - \mathbf{\Delta}_i \mathbf{\Delta}_i^\dagger \boldsymbol{\gamma}_i = \mathbf{0} \text{ if } j = i. \end{cases}$$

Proof of Property 5:

By symmetry of $\mathbf{H}$ it suffices to prove that $\langle \mathbf{H}\mathbf{r}_j, \mathbf{r}_i \rangle = 0$ for all $i > j$. This follows directly from $\langle \mathbf{H}\mathbf{r}_j, \mathbf{r}_i \rangle = \langle \mathbf{\Pi}\mathbf{H}\mathbf{r}_j, \mathbf{r}_i \rangle$ and Property 4 since $\mathbf{\Pi}\mathbf{H}\mathbf{r}_j \in \sum_{k=0}^{j} range(\mathbf{P}_k)$.

Proof of Property 2:

The minimization result is equivalent to the fact that $\mathbf{x}_i - \mathbf{x}_{00}$ is the $\mathbf{A}$-orthogonal projection of $\mathbf{x}_* - \mathbf{x}_{00}$ onto $range(\mathbf{U}) + \sum_{j=0}^{i-1} range(\mathbf{P}_j)$. With this, the proof comes down to the $\mathbf{A}$-orthogonality between this space and $\mathbf{x}_* - \mathbf{x}_i = (\mathbf{x}_* - \mathbf{x}_{00}) - (\mathbf{x}_i - \mathbf{x}_{00})$. For the space $range(\mathbf{U})$ it holds that:

$$\mathbf{U}^\top \mathbf{A}(\mathbf{x}_* - \mathbf{x}_i) = \mathbf{U}^\top \mathbf{r}_i = \mathbf{U}^\top \mathbf{\Pi}^\top \mathbf{r}_i = (\mathbf{\Pi}\mathbf{U})^\top \mathbf{r}_i = \mathbf{0} \text{ since } Ker(\mathbf{\Pi}) = range(\mathbf{U}).$$

For any of the spaces $range(\mathbf{P}_j)$ $(j = 0, \ldots, i - 1)$ the argument is Property 4: $\mathbf{P}_j^\top \mathbf{A}(\mathbf{x}_* - \mathbf{x}_i) = \mathbf{P}_j^\top \mathbf{r}_i = \mathbf{0}$.

Proof of Property 1:

The fact that $\mathbf{x}_{n-n_0} = \mathbf{x}_*$ follows from the observation that $rank(\mathbf{P}_i) \geq 1$ at every iteration until convergence is achieved. This is another way of saying that the

algorithm does not break down. Indeed, assume that $\text{rank}\,(\mathbf{P}_i) = 0$ then $\mathbf{\Pi H r}_i \in \text{span}\{\mathbf{P}_0, \ldots, \mathbf{P}_{i-1}\} = \text{span}\{\mathbf{\Pi H r}_0, \ldots, \mathbf{\Pi H r}_{i-1}\}$. Equivalently

$$\mathbf{H r}_i \in \text{span}\{\mathbf{H r}_0, \ldots, \mathbf{H r}_{i-1}\} + \text{Ker}(\mathbf{\Pi})$$
$$\Leftrightarrow \quad \mathbf{H}^{1/2}\mathbf{r}_i \in \text{span}\{\mathbf{H}^{1/2}\mathbf{r}_0, \ldots, \mathbf{H}^{1/2}\mathbf{r}_{i-1}\} + \mathbf{H}^{-1/2}\text{Ker}(\mathbf{\Pi}).$$

By Property 5 it holds that $\langle \mathbf{r}_i, \mathbf{H r}_j \rangle = 0$ for $j = 0, \ldots, i-1$ and $\mathbf{r}_i \in \text{range}(\mathbf{\Pi}^\top)$ so $\mathbf{r}_i \perp \text{Ker}(\mathbf{\Pi})$. It follows that if $\text{rank}\,(\mathbf{P}_i) = 0$, then the exact solution has been found before iteration $n - n_0$. If this hasn't occurred then, by a dimensional argument, at iteration $n - n_0$ the minimization space is the whole of $\mathbb{R}^n$ and $\mathbf{x}_{n-n_0} = \mathbf{x}_*$. $\square$

3.3. Convergence Results. The following theorems hold.

THEOREM 3.2. *If the exact solution has not yet been achieved at iteration $i - 1$ of Algorithm 2 and $t_{i-1} \geq \tau$ then the relative error is bounded by*

$$\frac{\|\mathbf{x}_* - \mathbf{x}_i\|_{\mathbf{A}}}{\|\mathbf{x}_* - \mathbf{x}_{i-1}\|_{\mathbf{A}}} \leq \left(\frac{1}{1 + \lambda_{\min}\tau} \right)^{1/2}.$$

(Recall that $\tau \in \mathbb{R}^+$ is the threshold chosen by the user and $\lambda_{\min}$ is a lower bound for the smallest eigenvalue of the preconditioned operator $\mathbf{HA}$.)

Proof. The proof follows the same lines of reasoning as the results in section 2. Recall the notation $\mathbf{d}_i = \mathbf{x}_* - \mathbf{x}_i$ which denotes the error at iteration i. By the finite termination property in Theorem 3.1 (Property 1), there exists an iteration number $I \leq n - n_0$ such that $\mathbf{x}_I = \mathbf{x}_*$ so $\mathbf{x}_* = \mathbf{x}_0 + \sum_{i=0}^{I-1} \mathbf{P}_i \boldsymbol{\alpha}_i = \mathbf{x}_i + \sum_{j=i}^{I-1} \mathbf{P}_j \boldsymbol{\alpha}_j$, or equivalently $\mathbf{d}_i = \sum_{j=i}^{I-1} \mathbf{P}_j \boldsymbol{\alpha}_j$. The blocks of search directions are pairwise $\mathbf{A}$-orthogonal (Property 3 in Theorem 3.1) so by subtraction $\|\mathbf{d}_i\|_{\mathbf{A}}^2 = \|\mathbf{d}_{i-1}\|_{\mathbf{A}}^2 - \|\mathbf{P}_{i-1}\boldsymbol{\alpha}_{i-1}\|_{\mathbf{A}}^2$. Then, recalling that $\|\mathbf{r}_i\|_H^2 = \|\mathbf{d}_i\|_{AHA}^2 \geq \lambda_{\min}\|\mathbf{d}_i\|_{\mathbf{A}}^2$ (by definition of $\lambda_{\min}$), it holds that:

$$\frac{\|\mathbf{d}_{i-1}\|_{\mathbf{A}}^2}{\|\mathbf{d}_i\|_{\mathbf{A}}^2} = 1 + \frac{\|\mathbf{P}_{i-1}\boldsymbol{\alpha}_{i-1}\|_{\mathbf{A}}^2}{\|\mathbf{r}_i\|_H^2} \frac{\|\mathbf{r}_i\|_H^2}{\|\mathbf{d}_i\|_{\mathbf{A}}^2} \geq 1 + \lambda_{\min}\frac{\|\mathbf{P}_{i-1}\boldsymbol{\alpha}_{i-1}\|_{\mathbf{A}}^2}{\|\mathbf{r}_i\|_H^2}.$$

The fraction corresponds to the quantity that is measured by the τ–test. Indeed

$$(3.1) \qquad \|\mathbf{P}_{i-1}\boldsymbol{\alpha}_{i-1}\|_{\mathbf{A}}^2 = \langle \boldsymbol{\gamma}_{i-1}, \boldsymbol{\Delta}_{i-1}^\dagger \boldsymbol{\Delta}_{i-1} \boldsymbol{\Delta}_{i-1}^\dagger \boldsymbol{\gamma}_{i-1} \rangle = \boldsymbol{\gamma}_{i-1}^\top \boldsymbol{\alpha}_{i-1},$$

so the assumption that $t_{i-1} \geq \tau$ can be rewritten as $\|\mathbf{P}_{i-1}\boldsymbol{\alpha}_{i-1}\|_{\mathbf{A}}^2 \geq \tau \|\mathbf{r}_i\|_H^2$ and the result follows. $\square$

The following corollary provides some insight into how to choose the parameter τ by choosing a targeted contraction factor ρ. This is related to the discussion that yielded (2.5) in Section 2.

COROLLARY 3.3. *Let $0 < \rho < 1$ and τ be chosen as $\tau = \frac{1-\rho^2}{\lambda_{min}\rho^2}$. Under the assumptions of Theorem 3.2 it holds that $\dfrac{\|\mathbf{x}_* - \mathbf{x}_i\|_{\mathbf{A}}}{\|\mathbf{x}_* - \mathbf{x}_{i-1}\|_{\mathbf{A}}} \leq \rho$.*

The next theorem states that if the problem is well-conditioned, then no adaptive step will be performed and the additional cost compared to PPCG is just the cost of performing the τ–test (two inner products and one scalar division) which is negligible.

THEOREM 3.4. *If all eigenvalues of the preconditioned operator $\mathbf{HA}$ are smaller than $1/\tau$ then the result of the τ–test is $t_i \geq \tau$ at each iteration and Algorithm 2 performs the usual PPCG iterations.*

Proof. Begin with $\mathbf{r}_i = \mathbf{r}_{i-1} - \mathbf{Q}_{i-1}\boldsymbol{\alpha}_{i-1}$ and take the inner product by $\mathbf{Hr}_i$:

$$\langle \mathbf{Hr}_i, \mathbf{r}_i \rangle = -\langle \mathbf{Hr}_i, \mathbf{AP}_{i-1}\boldsymbol{\alpha}_{i-1} \rangle \text{ (by Property 5 in Theorem 3.1)}.$$

An application of the Cauchy-Schwarz inequality in the $\mathbf{A}$-norm gives

$$\langle \mathbf{Hr}_i, \mathbf{r}_i \rangle \leq \langle \mathbf{Hr}_i, \mathbf{AHr}_i \rangle^{1/2} \langle \mathbf{P}_{i-1}\boldsymbol{\alpha}_{i-1}, \mathbf{AP}_{i-1}\boldsymbol{\alpha}_{i-1} \rangle^{1/2},$$

or equivalently

(3.2)
$$\frac{\langle \mathbf{Hr}_i, \mathbf{r}_i \rangle}{\langle \mathbf{Hr}_i, \mathbf{AHr}_i \rangle} \leq \frac{\langle \mathbf{P}_{i-1}\boldsymbol{\alpha}_{i-1}, \mathbf{AP}_{i-1}\boldsymbol{\alpha}_{i-1} \rangle}{\langle \mathbf{Hr}_i, \mathbf{r}_i \rangle}.$$

By assumption all eigenvalues of $\mathbf{HA}$ are smaller than $1/\tau$ so $\frac{\langle \mathbf{Hr}_i, \mathbf{r}_i \rangle}{\langle \mathbf{Hr}_i, \mathbf{AHr}_i \rangle} \geq \tau$ and by this, (3.2) and (3.1), the τ–test returns $t_i \geq \tau$. □

COROLLARY 3.5. *If the τ–test returns $t_{i-1} < \tau$ then $\|\mathbf{Hr}_i\|_{\mathbf{H}^{-1}}^2 < \tau\|\mathbf{Hr}_i\|_{\mathbf{A}}^2$. This implies that there is at least one eigenvalue of $\mathbf{HA}$ larger than $1/\tau$. Moreover, it holds that $\langle \mathbf{Hr}_i, \mathbf{Av} \rangle \neq 0$ where $\mathbf{v}$ is an eigenvector corresponding to that eigenvalue. This explains why it makes sense to augment the minimization space with the components of $\mathbf{Hr}_i$.*

Proof. The existence of an eigenvalue larger that $1/\tau$ follows easily from (3.2) and (3.1). □

3.4. Alternate Algorithm. One further assumption is made:

(A6) the operator $\mathbf{A}$ is a sum of N contributions : $\mathbf{A} = \sum\limits_{s=1}^{N} \mathbf{A}^s$, with each $\mathbf{A}^s$ symmetric and positive semidefinite.

In cases where the number N of components in the preconditioner is very large, it may occur that the cost of factorizing $\boldsymbol{\Delta}_i$ becomes excessive. In this case the *global* τ–test in Algorithm 2 (line 8) is switched with N *local* tests in order to deflate only the *local* components that are problematic. This is presented in Algorithm 3 and the adaptive steps are between lines 8 and 13.

The remarks from section 3.1 about the choice of τ, factorization of $\boldsymbol{\Delta}_i$ and full reorthogonalization also apply here. Additionally, if at a given iteration $\mathbf{H}^s\mathbf{r}_{i+1} = \mathbf{0}$, then t_i^s is not defined. It cannot occur that $\mathbf{H}^s\mathbf{r}_{i+1} = \mathbf{0}$ for all values of s unless convergence is achieved ($\mathbf{r}_{i+1} = \mathbf{0}$).

If the local τ–tests return $t_i^s < \tau$ for every $s = 1, \ldots, N$ at a given iteration, then $\mathbf{Z}_{i+1} = \left[\mathbf{Hr}_{i+1} | \mathbf{H}^1\mathbf{r}_{i+1} | \ldots | \mathbf{H}^N\mathbf{r}_{i+1}\right]$ and the first column is obviously linearly redundant so any efficient implementation of the algorithm would delete it immediately.

As is the case for the global τ–test, the evaluation of the local τ–tests relies on quantities that are available with little extra computational work. Indeed $\mathbf{Q}_i = \sum_{s=1}^{N} \mathbf{A}^s\mathbf{P}_i$, so $\mathbf{A}^s\mathbf{P}_i$ is available and a simple linear combination of its columns with the coefficients given by $\boldsymbol{\alpha}_i$ is performed, followed by the inner product with $\mathbf{P}_i\boldsymbol{\alpha}_i$. It makes sense to look for additional search directions locally if the preconditioner is constructed as a sum of approximate inverses $\mathbf{H}^s$ of the components $\mathbf{A}^s$ in the operator. This is illustrated with an example of substructuring (domain decomposition) solvers in the next section.

The theoretical properties of Algorithm 3 are stated in the following theorem and are similar to the ones of Algorithm 2.

THEOREM 3.6. *The five properties proved in Theorem 3.1 hold. The convergence bound in Theorem 3.2 holds if the local τ–tests return $t_i^s \geq \tau$ for every $s = 1, \ldots, N$ at a given iteration i.*

Proof. The proof of the first result is the same as the proof of Theorem 3.1. For the second result, the only additional argument is that $t_i^s \geq \tau$ can be rewritten as $\langle \mathbf{P}_i \boldsymbol{\alpha}_i, \mathbf{A}^s \mathbf{P}_i \boldsymbol{\alpha}_i \rangle \geq \tau \langle \mathbf{r}_{i+1}, \mathbf{H}^s \mathbf{r}_{i+1} \rangle$ and summing these estimates over $s = 1, \ldots, N$ gives $\langle \mathbf{P}_i \boldsymbol{\alpha}_i, \mathbf{A} \mathbf{P}_i \boldsymbol{\alpha}_i \rangle \geq \tau \langle \mathbf{H} \mathbf{r}_{i+1}, \mathbf{r}_{i+1} \rangle$. □

Algorithm 3: Adaptive multipreconditioned CG for $\left(\sum\limits_{s=1}^{N} \mathbf{A}^s \right) \mathbf{x}_* = \mathbf{b}$ preconditioned by $\left(\sum\limits_{s=1}^{N} \mathbf{H}^s \right)$ and $\mathbf{\Pi}$ for initial guess $\mathbf{x}_{00}$. $\tau \in \mathbb{R}^+$: chosen by user.

---
1 $\mathbf{x}_0 = \mathbf{\Pi} \mathbf{x}_{00} + \mathbf{U}(\mathbf{U}^\top \mathbf{A} \mathbf{U})^{-1} \mathbf{U}^\top \mathbf{b}; \ \mathbf{r}_0 = \mathbf{b} - \mathbf{A} \mathbf{x}_0; \ \mathbf{Z}_0 = \mathbf{H} \mathbf{r}_0; \ \mathbf{P}_0 = \mathbf{\Pi} \mathbf{Z}_0;$
2 **for** $i = 0, 1, \ldots,$ *convergence* **do**
3 $\quad \mathbf{Q}_i = \mathbf{A} \mathbf{P}_i;$
4 $\quad \mathbf{\Delta}_i = \mathbf{Q}_i^\top \mathbf{P}_i; \quad \boldsymbol{\gamma}_i = \mathbf{P}_i^\top \mathbf{r}_i; \quad \boldsymbol{\alpha}_i = \mathbf{\Delta}_i^\dagger \boldsymbol{\gamma}_i;$
5 $\quad \mathbf{x}_{i+1} = \mathbf{x}_i + \mathbf{P}_i \boldsymbol{\alpha}_i;$
6 $\quad \mathbf{r}_{i+1} = \mathbf{r}_i - \mathbf{Q}_i \boldsymbol{\alpha}_i;$ *(Locally) Adaptive Multipreconditioning*
7 $\quad \mathbf{Z}_{i+1} = \mathbf{H} \mathbf{r}_{i+1};$ // initialize $\mathbf{Z}_{i+1}$
8 $\quad$ **for** $s = 1, \ldots, N$ **do**
9 $\quad\quad t_i^s = \dfrac{\langle \mathbf{P}_i \boldsymbol{\alpha}_i, \mathbf{A}^s \mathbf{P}_i \boldsymbol{\alpha}_i \rangle}{\mathbf{r}_{i+1}^\top \mathbf{H}^s \mathbf{r}_{i+1}};$
10 $\quad\quad$ **if** $t_i^s < \tau$ **then** // local τ-test
11 $\quad\quad\quad \mathbf{Z}_{i+1} = [\mathbf{Z}_{i+1} \,|\, \mathbf{H}^s \mathbf{r}_{i+1}];$ // concatenate $\mathbf{Z}_{i+1}$ and $\mathbf{H}^s \mathbf{r}_{i+1}$
12 $\quad\quad$ **end**
13 $\quad$ **end**
14 $\quad \mathbf{\Phi}_{i,j} = \mathbf{Q}_j^\top \mathbf{Z}_{i+1}; \quad \boldsymbol{\beta}_{i,j} = \mathbf{\Delta}_j^\dagger \mathbf{\Phi}_{i,j}$ for each $j = 0, \ldots, i;$
15 $\quad \mathbf{P}_{i+1} = \mathbf{\Pi} \mathbf{Z}_{i+1} - \sum\limits_{j=0}^{i} \mathbf{P}_j \boldsymbol{\beta}_{i,j};$
16 **end**
17 Return $\mathbf{x}_{i+1};$

---

4. Application: Balancing Domain Decomposition (BDD). Domain Decomposition methods are linear solvers that are well-suited for parallel computing. They are hybrid solvers in the sense that they mix direct and iterative solves with the objective of achieving both robustness and parallel efficiency. The trick is that the domain is split into (sufficiently small) subdomains and all direct solves are performed inside these subdomains (where it is affordable) and not in the global domain. An iterative solver (*e.g.*, PPCG) connects the local components together. In this article, the focus is on one of the so called substructuring methods: Balancing Domain Decomposition, or BDD [31].

4.1. Notation and Introduction of the BDD formulation. Assume that a linear system $\mathbf{K} \mathbf{u} = \mathbf{f}$ is given for a symmetric positive definite matrix $\mathbf{K} \in \mathbb{R}^{m \times m}$ which corresponds to the finite element discretization of a partial differential equation (PDE) posed in an open subset Ω of $\mathbb{R}^2$ or $\mathbb{R}^3$. Further assume that Ω has been partitioned into N non overlapping and mesh conforming subdomains Ω^s and that Γ is the set of boundaries between subdomains:

$$\overline{\Omega} = \bigcup_{s=1,\ldots,N} \overline{\Omega^s}; \quad \Omega^s \cap \Omega^t = \emptyset \text{ for all } s \neq t; \quad \Gamma = \bigcup_{s=1,\ldots,N} \partial \Omega^s \setminus \partial \Omega.$$

The original linear system admits the following block form

$$\begin{pmatrix} \mathbf{K}_{\Gamma\Gamma} & \mathbf{K}_{\Gamma I} \\ \mathbf{K}_{I\Gamma} & \mathbf{K}_{II} \end{pmatrix} \begin{pmatrix} \mathbf{u}_{\Gamma} \\ \mathbf{u}_{I} \end{pmatrix} = \begin{pmatrix} \mathbf{f}_{\Gamma} \\ \mathbf{f}_{I} \end{pmatrix} \Leftrightarrow \begin{cases} \mathbf{K}_{\Gamma\Gamma}\mathbf{u}_{\Gamma} + \mathbf{K}_{\Gamma I}\mathbf{u}_{I} = \mathbf{f}_{\Gamma} \\ \mathbf{K}_{I\Gamma}\mathbf{u}_{\Gamma} + \mathbf{K}_{II}\mathbf{u}_{I} = \mathbf{f}_{I} \end{cases},$$

where the subscript $*_{\Gamma}$ denotes the restriction to the set of degrees of freedom on Γ and $*_{I}$ to the remaining unknowns. From the second line it can be deduced that $\mathbf{u}_{I} = \mathbf{K}_{II}^{-1}(\mathbf{f}_{I} - \mathbf{K}_{I\Gamma}\mathbf{u}_{\Gamma})$ and by injecting this into the first line the problem is reduced to one on the interfaces between subdomains: Find $\mathbf{u}_{\Gamma} \in \mathbb{R}^{n}$ ($n := \#(\Gamma)$) such that

$$(4.1) \qquad \mathbf{A}\mathbf{u}_{\Gamma} = \mathbf{b}, \text{ where } \mathbf{A} := \mathbf{K}_{\Gamma\Gamma} - \mathbf{K}_{\Gamma I}\mathbf{K}_{II}^{-1}\mathbf{K}_{I\Gamma} \text{ and } \mathbf{b} := \mathbf{f}_{\Gamma} - \mathbf{K}_{\Gamma I}\mathbf{K}_{II}^{-1}\mathbf{f}_{I}.$$

This is the linear system that is solved in the BDD method. The result is an approximation $\mathbf{u}_{\Gamma}$ of the solution on Γ and the remaining part of the solution is computed as $\mathbf{u}_{I} = \mathbf{K}_{II}^{-1}(\mathbf{f}_{I} - \mathbf{K}_{I\Gamma}\mathbf{u}_{\Gamma})$. To understand why BDD is ideal for parallel computing and fits the framework for adaptive multipreconditioning, (4.1) is rewritten into a form that makes the local contributions apparent. First, let $\mathbf{K}^{s}$ be the local matrices corresponding to the discretization of the same PDE but restricted to each subdomain $\overline{\Omega^{s}}$ and write them in block formulation as

$$\mathbf{K}^{s} = \begin{pmatrix} \mathbf{K}_{\Gamma^{s}\Gamma^{s}}^{s} & \mathbf{K}_{\Gamma^{s}I^{s}}^{s} \\ \mathbf{K}_{I^{s}\Gamma^{s}}^{s} & \mathbf{K}_{I^{s}I^{s}}^{s} \end{pmatrix} \text{ where } \begin{cases} *_{\Gamma^{s}}: \text{ degrees of freedom on } \Gamma \cap \partial\Omega^{s}, \\ *_{I^{s}}: \text{ remaining degrees of freedom in } \overline{\Omega^{s}}. \end{cases}$$

Then, define the local Schur complements $\mathbf{S}^{s} := \mathbf{K}_{\Gamma^{s}\Gamma^{s}}^{s} - \mathbf{K}_{\Gamma^{s}I^{s}}^{s}(\mathbf{K}_{I^{s}I^{s}}^{s})^{-1}\mathbf{K}_{I^{s}\Gamma^{s}}^{s}$. Finally, these can be assembled into the BDD operator already defined in (4.1). Indeed it holds that

$$(4.2) \qquad \mathbf{A} = \sum_{s=1}^{N} \mathbf{A}^{s}, \text{ where for all } s = 1, \dots, N: \quad \mathbf{A}^{s} := \mathbf{R}^{s\top}\mathbf{S}^{s}\mathbf{R}^{s}$$

and $\mathbf{R}^{s} \in \mathbb{R}^{\#(\Gamma^{s})\times n}$ is the boolean matrix that, given a vector in $\mathbb{R}^{n}$, selects the entries in Γ^{s}. The fact that $\mathbf{A}$ is a sum of local contributions has now been made apparent and the preconditioner exploits this since it is

$$(4.3) \qquad \mathbf{H} := \sum_{s=1}^{N} \mathbf{H}^{s}, \text{ where for all } s = 1, \dots, N: \quad \mathbf{H}^{s} := \mathbf{R}^{s\top}\mathbf{D}^{s}\mathbf{S}^{s\dagger}\mathbf{D}^{s}\mathbf{R}^{s},$$

and $\{\mathbf{D}^{s}\}_{s=1,\dots,N}$ is a family of positive definite diagonal matrices that form a partition of unity (i.e., they satisfy $\sum_{s=1}^{N}\mathbf{R}^{s\top}\mathbf{D}^{s}\mathbf{R}^{s} = \mathbf{I}$). Once more, $\dagger$ denotes a pseudo inverse. This last piece of notation reveals a difficulty inherent to BDD: if the local problems $\mathbf{K}^{s}$ are not positive definite then neither are the Schur complements $\mathbf{S}^{s}$. This difficulty, observed in [31], has since been overcome by adding a deflation step to the Neumann Neumann algorithm in [6]. The deflation, or *coarse*, space is chosen as:

$$(4.4) \qquad \text{range}(\mathbf{U}) = \sum_{s=1}^{N} \mathbf{R}^{s\top}\mathbf{D}^{s}\text{Ker}(\mathbf{S}^{s}).$$

This offers the double advantage of making all applications of $\mathbf{\Pi}\mathbf{H}$ and $\mathbf{H}\mathbf{\Pi}^{\top}$ uniquely defined (regardless of the choice of the pseudo inverse) and of improving convergence

significantly. An alternative approach is the Balancing Domain Decomposition by Constraints (or BDDC) solver [7].

REMARK 5 (Singularity of $\mathbf{S}^s$: an example). *If the PDE underlying the linear system is: find u satisfying some boundary conditions such that $-\Delta u = f$ (for a right hand side f) then $\mathbf{K}^s$ is the matrix associated with the discretization of the variational form $(u,v) \mapsto \int_{\Omega^s} \nabla u \cdot \nabla v$. It is obvious that all constant functions are in its kernel. This in turn induces the singularity of the local matrices $\mathbf{K}^s$ and of their Schur complements $\mathbf{S}^s$.*

4.2. New Adaptive BDD. There were six assumptions in sections 2 and 3. With the notation introduced in (4.2), (4.3) and (4.4) all six of these assumptions hold: $\mathbf{U}$ is a full rank matrix, $\mathbf{A}$ and $\mathbf{H}$ are symmetric positive definite matrices [31] assembled as the sum of N symmetric positive semidefinite matrices and all eigenvalues of the preconditioned operator $\mathbf{HA}$ are larger than $\lambda_{\min} = 1$ [31, 49]. As a result, the two new adaptive MPCG algorithms (namely, Algorithm 2 with the global τ–test, and Algorithm 3 with the local τ–test) can straightforwardly be applied to the BDD linear system (4.1). Moreover, the theoretical results in Theorems 3.1, 3.2, 3.4, 3.6 and Corollary 3.5 hold.

It is well known that the majority of time and resource consuming operations in a BDD algorithm are the local solves required by any application of the operator $\mathbf{A}$ and the preconditioner $\mathbf{H}$ (specifically, the Dirichlet solves $\mathbf{K}^s_{I^s I^s}{}^{-1}$ in $\mathbf{S}^s$ and the Neumann solves $\mathbf{K}^{s\dagger}$ used to compute an application of $\mathbf{S}^{s\dagger}$). We observe that the cost of preconditioning is the same in one iteration of adaptive MPCG as in an iteration of PPCG for BDD. However, in iterations where multiple search directions are selected, additional applications of $\mathbf{A}$ are needed if the original formulation of the algorithms is implemented ($\mathbf{P}_i$ is dense after orthogonalization and projection). Since the interest is in high performance computing Algorithms 4 and 5 are proposed, which are optimized versions of the algorithms for BDD. In exact arithmetic the modifications make no difference to the sequence of approximate solutions but they save a lot of computational time.

Following the trick in [16] (equation (10)) it is ensured that all additional applications of $\mathbf{A}$ are performed on vectors that are supported in one subdomain, meaning that they only require a local solve in that particular subdomain and its neighbours. In the numerical results section, PPCG, MPCG, and the adaptive MPCG algorithms are compared in terms of the number of local solves, for which it is observed that the newly proposed adaptive MPCG solvers are extremely competitive.

In line 3 of Algorithm 4 it is proposed to compute $\tilde{\mathbf{Q}}_i = \mathbf{AZ}_i - \sum_{j=0}^{i-1} \tilde{\mathbf{Q}}_j \boldsymbol{\beta}_{i-1,j}$ instead of $\mathbf{Q}_i = \mathbf{AP}_i$. The connection is that $\mathbf{Q}_i = \mathbf{\Pi}\tilde{\mathbf{Q}}_i$. The values of $\boldsymbol{\Delta}_i$ and all the $\boldsymbol{\Phi}_{i,j}$ remain the same. One more application of the projection $\mathbf{\Pi}^\top$ to a vector is performed in line 6, which is a negligible cost compared to the significant gain that results from the localization of the applications of $\mathbf{A}$.

In Algorithm 5, the formulation is slightly more complex since the local components $\mathbf{Q}^s_i = \mathbf{A}^s \mathbf{P}_i$ are also explicitly computed in line 4 to perform the local τ–test:

(4.5)
$$\mathbf{Q}^s_i = \mathbf{A}^s \left(\mathbf{\Pi}\mathbf{Z}_i - \sum_{j=0}^{i-1} \mathbf{P}_j \boldsymbol{\beta}_{i-1,j} \right) = \mathbf{A}^s \mathbf{Z}_i - \mathbf{A}^s \mathbf{U}(\mathbf{U}^\top \mathbf{A}\mathbf{U})^{-1}(\mathbf{A}\mathbf{U})^\top \mathbf{Z}_i - \sum_{j=0}^{i-1} \mathbf{Q}^s_j \boldsymbol{\beta}_{i-1,j}.$$

Again, this allows for a significant reduction in the number of local solves that are incurred by applying $\mathbf{A}$ each time some local contributions are selected (when the τ–test returns $t_i^s < \tau$). The second term in the sum is not very costly to compute since it consists of some low rank corrections. An additional modification is that in line 13 the first column in $\mathbf{Z}_i$ is made sparser by subtracting local contributions from the global vector when they contribute independently to the minimization space. This also reduces the number of local solves. Finally, in order to save some projection steps $\tilde{\mathbf{P}}_i$ is computed instead of $\mathbf{P}_i = \mathbf{\Pi}\tilde{\mathbf{P}}_i$, with the consequence that in line 6 the projection $\mathbf{\Pi}$ is applied to a vector.

Algorithm 4: Algorithm 2 for BDD: Adaptive MPCG with global τ–test

---

1 $\mathbf{x}_0 = \mathbf{\Pi}\mathbf{x}_{00} + \mathbf{U}(\mathbf{U}^\top\mathbf{A}\mathbf{U})^{-1}\mathbf{U}^\top\mathbf{b}$; $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$; $\mathbf{Z}_0 = \mathbf{H}\mathbf{r}_0$; $\mathbf{P}_0 = \mathbf{\Pi}\mathbf{Z}_0$;

2 **for** $i = 0, 1, \ldots,$ *convergence* **do**

3     $\quad \tilde{\mathbf{Q}}_i = \mathbf{A}\mathbf{Z}_i - \sum\limits_{j=0}^{i-1} \tilde{\mathbf{Q}}_j \boldsymbol{\beta}_{i-1,j}$;

4     $\quad \boldsymbol{\Delta}_i = \tilde{\mathbf{Q}}_i^\top\mathbf{P}_i$; $\quad \boldsymbol{\gamma}_i = \mathbf{P}_i^\top\mathbf{r}_i$; $\quad \boldsymbol{\alpha}_i = \boldsymbol{\Delta}_i^\dagger\boldsymbol{\gamma}_i$;

5     $\quad \mathbf{x}_{i+1} = \mathbf{x}_i + \mathbf{P}_i\boldsymbol{\alpha}_i$;

6     $\quad \mathbf{r}_{i+1} = \mathbf{\Pi}^\top(\mathbf{r}_i - (\tilde{\mathbf{Q}}_i\boldsymbol{\alpha}_i))$; $\qquad\qquad$ *Adaptive Multipreconditioning*

7     $\quad t_i = \dfrac{\boldsymbol{\gamma}_i^\top\boldsymbol{\alpha}_i}{\mathbf{r}_{i+1}^\top\mathbf{H}\mathbf{r}_{i+1}}$;

8     $\quad$ **if** $t_i < \tau$ **then** $\qquad\qquad\qquad\qquad\qquad\qquad$ // global τ-test

9     $\quad\quad\mid \quad \mathbf{Z}_{i+1} = \left[\mathbf{H}^1\mathbf{r}_{i+1}\mid \ldots \mid \mathbf{H}^N\mathbf{r}_{i+1}\right]$;

10    $\quad$ **else**

11    $\quad\quad\mid \quad \mathbf{Z}_{i+1} = \mathbf{H}\mathbf{r}_{i+1}$;

12    $\quad$ **end**

13    $\quad \boldsymbol{\Phi}_{i,j} = \tilde{\mathbf{Q}}_j^\top\mathbf{\Pi}\mathbf{Z}_{i+1}$; $\quad \boldsymbol{\beta}_{i,j} = \boldsymbol{\Delta}_j^\dagger\boldsymbol{\Phi}_{i,j}$ $\quad$ for each $j = 0, \ldots, i$;

14    $\quad \mathbf{P}_{i+1} = \mathbf{\Pi}\,\mathbf{Z}_{i+1} - \sum\limits_{j=0}^{i} \mathbf{P}_j\boldsymbol{\beta}_{i,j}$;

15 **end**

16 Return $\mathbf{x}_{i+1}$;

4.3. Connection with previous work. As previously mentioned in section 2.2 it makes sense to generate the deflation space range($\mathbf{U}$) with the eigenvectors corresponding to isolated eigenvalues of the preconditioned operator $\mathbf{H}\mathbf{A}$. Since it is too costly to compute these eigenvectors, the authors in [33, 44, 47, 45, 26] instead proposed approximating this space with local contributions. More precisely in [47, Theorem 2.11 and Remark 2.9], given a threshold $\tau \in \mathbb{R}^+$, the coarse space is chosen so that

$$(4.6) \qquad \text{range}(\mathbf{U}) = \text{span}\{\mathbf{R}^{s\top}\mathbf{p}_k^s; s = 1, \ldots, N \text{ and } \lambda_k^s \leq \tau\},$$

where in each subdomain $(\mathbf{p}_k^s, \lambda_k^s) \in (\mathbb{R}^{\#(\Gamma^s)}, \mathbb{R}^+)$ are the eigenpairs of the following generalized eigenvalue problem

$$(\mathbf{D}^{s-1}\mathbf{S}^s\mathbf{D}^{s-1})\,\mathbf{p}_k^s = \lambda_k^s\,(\mathbf{R}^s\mathbf{A}\mathbf{R}^{s\top})\,\mathbf{p}_k^s.$$

Algorithm 1 is then applied. It is proven that with this choice of $\mathbf{U}$ the largest eigenvalue of the projected preconditioned operator $\mathbf{H}\mathbf{A}\mathbf{\Pi}$ is bounded by $\mathcal{N}/\tau$ where

---

Algorithm 5: Algorithm 3 for BDD: Adaptive MPCG with local τ–test

---

1 $\mathbf{x}_0 = \mathbf{\Pi}\mathbf{x}_{00} + \mathbf{U}(\mathbf{U}^\top\mathbf{A}\mathbf{U})^{-1}\mathbf{U}^\top\mathbf{b}$; $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$; $\mathbf{Z}_0 = \mathbf{H}\mathbf{r}_0$;

2 $\tilde{\mathbf{P}}_0 = \mathbf{Z}_0$;

3 **for** $i = 0, 1, \ldots,$ *convergence* **do**

4 $\quad$ $\mathbf{Q}_i = \sum_{s=1}^N \mathbf{Q}_i^s$ where

$\quad\quad$ $\mathbf{Q}_i^s = \mathbf{A}^s\mathbf{Z}_i - \mathbf{A}^s\mathbf{U}(\mathbf{U}^\top\mathbf{A}\mathbf{U})^{-1}(\mathbf{A}\mathbf{U})^\top\mathbf{Z}_i - \sum_{j=0}^{i-1} \mathbf{Q}_j^s\boldsymbol{\beta}_{i-1,j}$;

5 $\quad$ $\boldsymbol{\Delta}_i = \mathbf{Q}_i^\top\tilde{\mathbf{P}}_i$; $\quad$ $\boldsymbol{\gamma}_i = \tilde{\mathbf{P}}_i^\top\mathbf{r}_i$; $\quad$ $\boldsymbol{\alpha}_i = \boldsymbol{\Delta}_i^\dagger\boldsymbol{\gamma}_i$;

6 $\quad$ $\mathbf{x}_{i+1} = \mathbf{x}_i + \mathbf{\Pi}\tilde{\mathbf{P}}_i\boldsymbol{\alpha}_i$;

7 $\quad$ $\mathbf{r}_{i+1} = \mathbf{r}_i - \mathbf{Q}_i\boldsymbol{\alpha}_i$; $\quad\quad\quad$ *(Locally) Adaptive Multipreconditioning*

8 $\quad$ $\mathbf{Z}_{i+1} = \mathbf{H}\mathbf{r}_{i+1}$;

9 $\quad$ **for** $s = 1, \ldots, N$ **do**

10 $\quad\quad$ $t_i^s = \dfrac{\langle\mathbf{\Pi}\tilde{\mathbf{P}}_i\boldsymbol{\alpha}_i, \mathbf{Q}_i^s\boldsymbol{\alpha}_i\rangle}{\mathbf{r}_{i+1}^\top\mathbf{H}^s\mathbf{r}_{i+1}}$;

11 $\quad\quad$ **if** $t_i^s < \tau$ **then** $\quad\quad\quad\quad\quad\quad\quad\quad\quad$ // local τ-test

12 $\quad\quad\quad$ $\mathbf{Z}_{i+1} = [\mathbf{Z}_{i+1} \,|\, \mathbf{H}^s\mathbf{r}_{i+1}]$;

13 $\quad\quad\quad$ Subtract $\mathbf{H}^s\mathbf{r}_{i+1}$ from the first column in $\mathbf{Z}_{i+1}$;

14 $\quad\quad$ **end**

15 $\quad$ **end**

16 $\quad$ $\boldsymbol{\Phi}_{i,j} = \mathbf{Q}_j^\top\mathbf{Z}_{i+1}$; $\quad$ $\boldsymbol{\beta}_{i,j} = \boldsymbol{\Delta}_j^\dagger\boldsymbol{\Phi}_{i,j}$ $\quad$ for each $j = 0, \ldots, i$;

17 $\quad$ $\tilde{\mathbf{P}}_{i+1} = \mathbf{Z}_{i+1} - \sum_{j=0}^i \tilde{\mathbf{P}}_j\boldsymbol{\beta}_{i,j}$;

18 **end**

19 Return $\mathbf{x}_{i+1}$;

N is the maximal number of neighbours of a subdomain (including itself). Since $\lambda_{\min}$ is still 1, the classical estimate (2.2) yields $\frac{\|\mathbf{x}_* - \mathbf{x}_i\|_\mathbf{A}}{\|\mathbf{x}_* - \mathbf{x}_0\|_\mathbf{A}} \leq 2\left[\frac{\sqrt{N/\tau} - 1}{\sqrt{N/\tau} + 1}\right]^i$. This result indicates that with local contributions it is possible to guarantee any targeted convergence rate by adjusting τ. Although this can be done in parallel, and only a few eigenvectors are needed, the fact that the setup requires solving generalized eigenvalues problems is a drawback. The original motivation for the present work was to achieve fast convergence with a coarse space that is enriched on the fly. Even though the τ–test in the proposed algorithms does not exactly measure the same quantities as the generalized eigenvalue problem, it is expected to be able to capture the relevant quantities within the iterations of the two adaptive MPCG algorithms and ensure fast convergence. We will compare both approaches in the next section.

REMARK 6. *The theoretical result for the adaptive MPCG algorithms is that at iterations where the τ–test returns $t_i \geq \tau$ (respectively, $t_i^s \geq \tau$ for all s in the case of the local test) it is guaranteed that the error decreases linearly. However it is not proven how often this happens. This is similar to the GenEO result since in that case no result is given for the size of the coarse space. These questions are beyond the scope of this paper and are not considered further here.*

5. Numerical Results. The results in this section were obtained with the software libraries FreeFem++ [21] and GNU Octave [10]. The linear elasticity equations

posed in $\Omega = [0,1]^2$ with mixed boundary conditions are considered. A solution $\mathbf{u} = (u_1, u_2)^\top \in H^1(\Omega)^2$ is sought such that

(5.1)
$$
\begin{cases}
-\mathrm{div}(\sigma(\mathbf{u})) = (0,10)^\top, & \text{in } \Omega, \\
\qquad\qquad \mathbf{u} = (0,0)^\top, & \text{on } \{(x,y) \in \partial\Omega : x = 0\}, \\
\qquad\quad \sigma(\mathbf{u}) \cdot \mathbf{n} = 0, & \text{on the rest of } \partial\Omega \text{ (with } \mathbf{n} \text{ denoting the outward normal).}
\end{cases}
$$

The stress tensor $\sigma(\mathbf{u})$ is defined by $\sigma_{ij}(\mathbf{u}) = 2\mu\varepsilon_{ij}(\mathbf{u}) + \lambda\delta_{ij}\mathrm{div}(\mathbf{u})$ for $i, j = 1, 2$ where $\varepsilon_{ij}(\mathbf{u}) = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right)$, δ_{ij} is the Kronecker symbol and the Lamé coefficients are functions of Young's modulus E and Poisson's ratio ν : $\mu = \frac{E}{2(1+\nu)}$, $\lambda = \frac{E\nu}{(1+\nu)(1-2\nu)}$. It is well known (see, e.g., [39]) that the solution of (5.1) in a heterogeneous medium is challenging due to ill-conditioning. In the following experiments, after setting $\nu = 0.4$ in all of the domain for all test cases, a checkerboard distribution is chosen for E (see Figure 1c) where the two values are $E_1 = 10^7$ and, unless specified otherwise, $E_2 = 10^{12}$.

The computational domain is discretized by a uniform mesh and the boundary value problem is solved numerically with standard piecewise linear ($\mathbb{P}_1$) Lagrange finite elements. The resulting linear system is then rewritten in BDD formulation. To this end the computational domain is partitioned into N subdomains (N is given). The partition is either regular (into squares of size $1/\sqrt{N} \times 1/\sqrt{N}$) or generated automatically by the graph partitioner Metis [25] (see Figures 1a and 1b). The mesh parameter is inversely proportional to the number of subdomains: $h = 1/(10N)$. The material in each subdomain is either homogeneous (if the partition is regular as in Section 5.1) or heterogeneous (if the partition is computed by Metis as in Section 5.2).

Five algorithms for solving the linear system are compared: Algorithm 4 (Adaptive MPCG with Global τ–test), Algorithm 5 (Adaptive MPCG with Local τ–test), Algorithm 4 with $\tau = \infty$ (Simultaneous), Algorithm 1 where $\mathbf{U}$ is full rank and satisfies (4.4) (PPCG) and Algorithm 1 where $\mathbf{U}$ is full rank and satisfies (4.6) (GenEO). The stopping criterion is always that the error $\|\mathbf{x}_i - \mathbf{x}_*\|_{\mathbf{A}}$ be smaller than $10^{-6}\|\mathbf{x}_*\|_{\mathbf{A}}$. This allows us to compare all algorithms fairly. Finally, for all three adaptive methods (both new adaptive MPCG methods, *cf.* Algorithms 4 and 5, and GenEO) $\tau = 0.1$ is chosen except in one explicitly stated test, where τ is varied. According to Corollary 3.3 (with $\lambda_{min} = 1$ for BDD) this guarantees that, at iterations where only one search direction is used, the error has decreased by at least a factor $\rho = 0.95$.

We examine three quantities with respect to the iteration count: the relative error $\|\mathbf{x}_i - \mathbf{x}_*\|_{\mathbf{A}}/\|\mathbf{x}_*\|_{\mathbf{A}}$, the dimension of the minimization space (see Property 2 in Theorem 3.1), and the number of local solves (number of applications of $\mathbf{S}^s$ or $\mathbf{S}^{s\dagger}$). In the case of GenEO, results on the number of local solves are never included since this would also require counting the number of iterations needed for the eigensolver to converge and optimizing this is not the purpose of this article. We leave the study in terms of CPU time for future work as the scope of this article is to illustrate the theoretically predicted behaviour of the algorithms.

5.1. Regular partition: Homogeneous Subdomains. The partition is into $N = 81$ subdomains (see Figure 1a) in such a way that the material parameters are constant in each subdomain. The size of the linear system is $n = 3056$. We consider two choices for the partition of unity matrices $\mathbf{D}^s$ in the preconditioner. Either $\mathbf{D}^s$ is proportional to the diagonal values of the local matrix $\mathbf{K}^s$ (k-scaling) [32, 41, 29, 39] or it is equal to the inverse of the multiplicity of the degrees of freedom (multiplicity scaling).

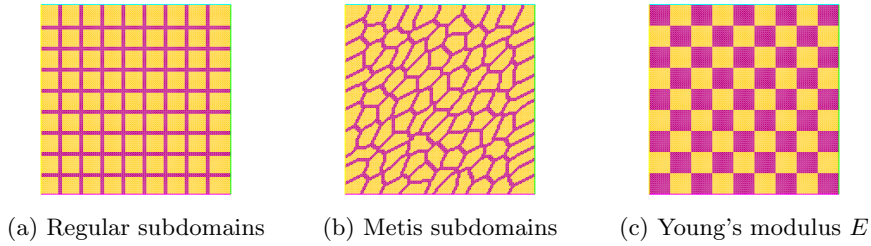(a) Regular subdomains   (b) Metis subdomains   (c) Young's modulus E

Fig. 1: Geometry of the test problem. There are 19800 degrees of freedom in the original problem and 3056 (respectively, 3458) in its BDD formulation on the regular (respectively, Metis) partition into 81 subdomains.

Figure 2a shows the results when k-scaling is used. Theoretical results [31, 29, 49] guarantee that PPCG converges fast in this case. We observe that this is indeed the case and that the adaptive algorithms detect that they do not need to do any extra work (or very little in the case of adaptive MPCG with the local τ–test which selects just 4 extra search directions). The simultaneous algorithm converges even faster but at the cost of about twice as many local solves.

When multiplicity scaling is used, see Figure 2b, convergence of PPCG is not as good. Both of the adaptive MPCG algorithms adapt by selecting significantly larger minimization spaces (626 and 783 versus 268). This allows for a reduction in the number of iterations from over 50 to under 10 and the number of local solves decreases by about a factor 2 from 8586 to 4302 and 4176. The algorithm with the global τ–test augments the minimization space at each iteration so in this case it is the same as the Simultaneous algorithm. The coarse space computed by GenEO was prohibitively large so GenEO was not included.

As predicted, in each case the adaptive MPCG algorithms adapt correctly to the difficulty of the problem: for the harder problem, convergence is (almost) as good as with the Simultaneous algorithm, and for the easier problem (almost) no extra work is performed compared to PPCG. Next a more complex geometry is considered for which neither the Simultaneous Algorithm or PPCG are expected to be optimal.

5.2. Metis partition: Heterogeneous Subdomains. We partition the domain into $N = 81$ subdomains with Metis (see Figure 1b). This way, the partition into subdomains is no longer aligned with the heterogeneities and the domains are non regular. Subdomains with these properties are known to cause problems for BDD even with k-scaling (which is used here). The size of the linear system for this problem is $n = 3458$.

We first plot the results for all test cases in Figure 3a, and then without PPCG in Figure 3b in order to highlight the differences between the competing methods. The convergence history of PPCG (first plot in Figure 3a) illustrates the typical behaviour of conjugate gradient algorithms for ill-conditioned systems. For all the other methods the stagnation is completely eliminated (GenEO) or significantly decreased. This confirms that the problematic eigenvectors are well-handled by augmenting the minimization space with local quantities. Observe that with the adaptive MPCG algorithms the minimization space is augmented only during the first few iterations. Then they behave as fast converging PPCG methods and in the end they require roughly
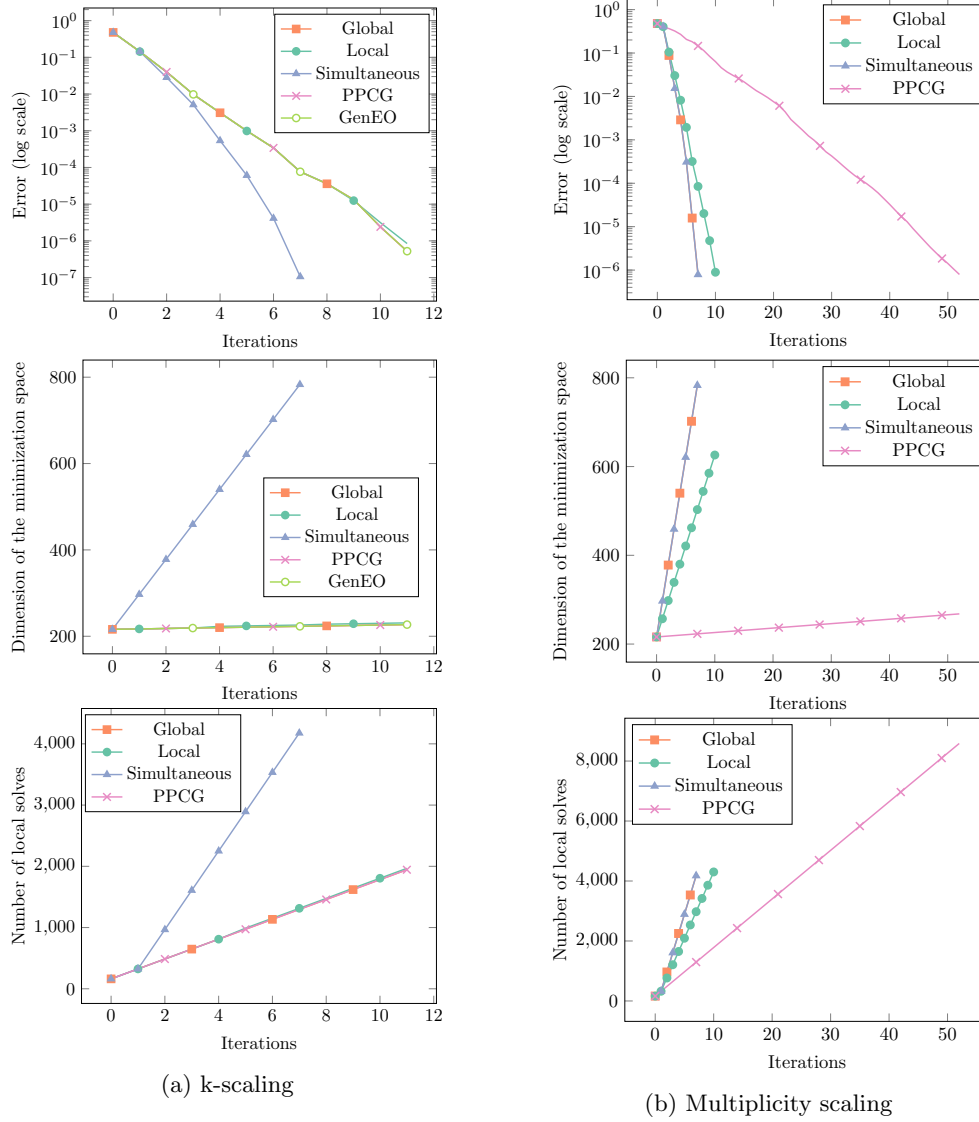
(a) k-scaling

(b) Multiplicity scaling

Fig. 2: Regular partition: the subdomains are homogeneous.

4 times fewer local solves than PPCG. For this harder test case, the Simultaneous algorithm also converges faster and with fewer local solves than PPCG.

Influence of τ. The threshold τ plays a very important role in the adaptive MPCG algorithms. The global $(t_i < \tau)$ or local $(t_i^s < \tau)$ τ-tests determine whether to perform an iteration with one or several search directions, respectively. In Corollary 3.3, it is shown that τ can be related to a targeted contraction factor ρ that has a more natural interpretation. We have performed the simulation with both adaptive MPCG algorithms for the same test problem as previously but with values of τ ranging between 10^{-4} and 10. In Figure 4, the contraction factor ρ given by Corollary 3.3

is plotted, as well as the iteration count, dimension of the minimization space, and number of local solves required by each algorithm (all as functions of τ). For the method with the global τ-test, there is a whole range of values ($0.06 \leq \tau \leq 0.4$ or equivalently $0.846 \leq \rho \leq 0.997$) which is optimal in the sense of minimizing the number of local solves. This illustrates that the method is robust with respect to variations in τ. In fact, for all values of τ considered, the adaptive MPCG algorithms outperformed both non adaptive algorithms: PPCG (22842 local solves) and the Simultaneous algorithm (8360 local solves). One (heuristic) explanation as to why there are several *good* choices for τ is that there are two regimes in the convergence history of PPCG (first plot in Figure 3a): a long stagnation during the first 100 or so iterations followed by fast convergence once the large isolated eigenvalues of the preconditioned BDD operator have been well approximated by the Ritz values [50]. The transition between both regimes is quite brutal and that is why the best choice of τ is not unique.

For the local version of the algorithm, the minimal number of local solves is 4743 instead of 5212 with the global τ-test. As expected, the minimization space is smaller than with the global version of the algorithm but sometimes this leads to a larger iteration count and more local solves overall. In the end the local version of the algorithm appears to be better for the best choice of τ but also more sensitive to the choice of τ.

Influence of E_2/E_1 and N. In Table 1, the influence of other problem parameters on all methods for the same test case with $\tau = 0.1$ is presented. First, details on the number of iterations and local solves needed for convergence when the heterogeneity E_2/E_1 varies (both for k-scaling and multiplicity scaling) are given. Then the ratio is set back to $E_2/E_1 = 10^5$ and the number N of Metis subdomains is varied. Note that the number of squares in the checkerboard is also N and the mesh parameter is $h = 1/(10N)$. The last line of each table gives the maximal size of the coarse space for that column. As a general rule, only PPCG suffers in terms of number of iterations when the problems become harder. This is important because each iteration requires a global communication step which is a bottleneck in terms of parallelism. For all methods, the number of local solves becomes larger as the problems become harder but the impact is a lot less important for the two adaptive MPCG algorithms illustrating that they adapt well to more difficult problems. One final argument in favour of the adaptive MPCG algorithms is that block operations are often proportionally much less expensive than single vector operations because the computation time is driven by the memory access. Note that this point was previously made in [16] for the Simultaneous algorithm.

6. Conclusion and Perspectives. In this article, a new adaptive conjugate gradient solver is proposed for linear systems with multiple preconditioners (as well as a variant). The theoretical analysis guarantees that there are two kinds of iterations: either the error is reduced by the targeted convergence factor and a standard PPCG iteration is taken, or otherwise, the minimization space is enriched with components coming from several different preconditioners. Good convergence was observed when applying the solver to the system arising from Balancing Domain Decomposition. The adaptive MPCG algorithms should constitute good parallel solvers for industrial applications.

In the future the algorithms will be tested on realistic test cases with optimized parallel implementations. Future work should also include applying the algorithm to other linear systems such as ones with multiple preconditioners that arise from

(a) Results for all methods
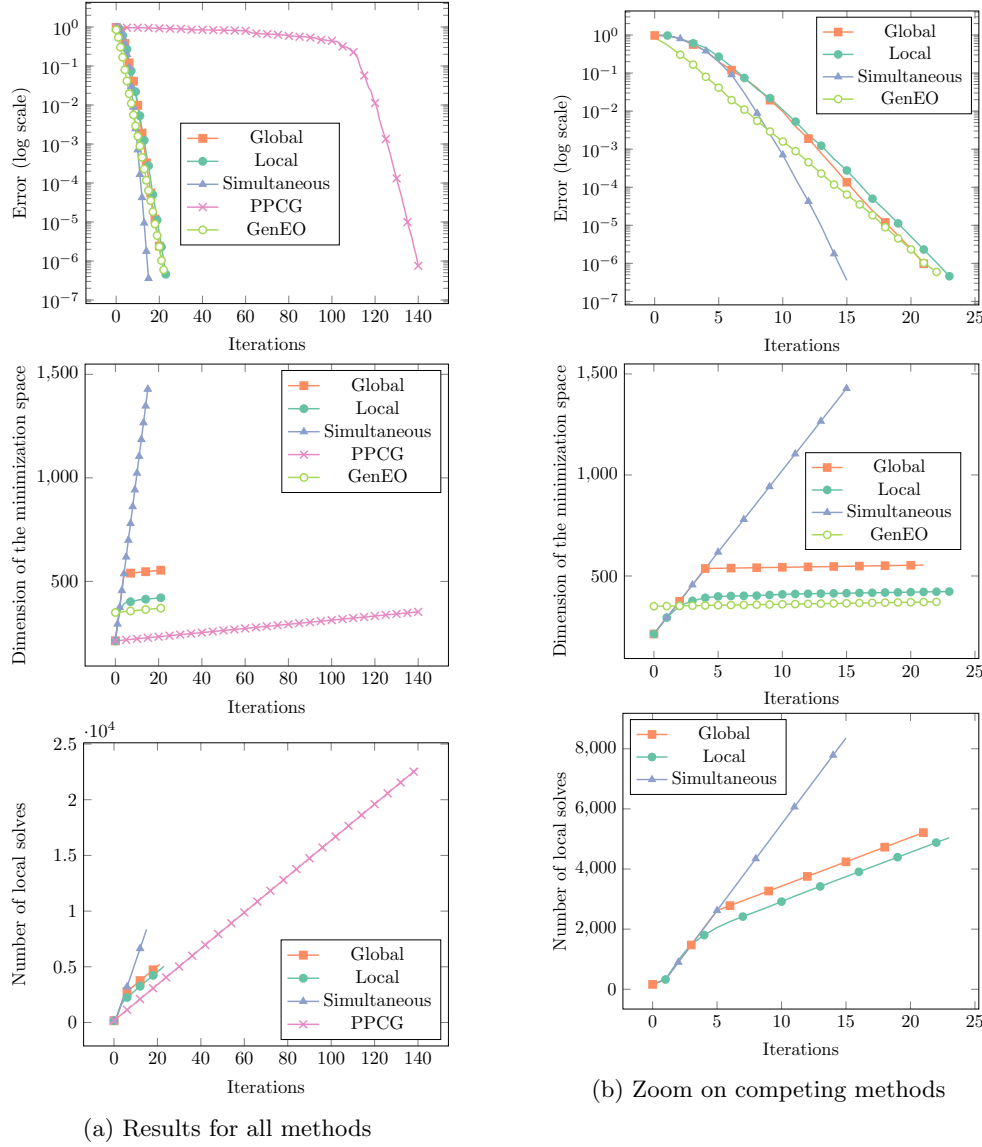
(b) Zoom on competing methods

Fig. 3: Metis Partition : the subdomains are heterogeneous.

the physics of the problem. Another objective would be to consider the case where a bound for the largest eigenvalue is known instead of the smallest (or neither is available). In this case the algorithm would apply also to the Additive Schwarz and Optimized Schwarz domain decomposition methods. Finally generalization to non symmetric problems should be considered, this will require generalizing the ideas from this paper to other Krylov methods.
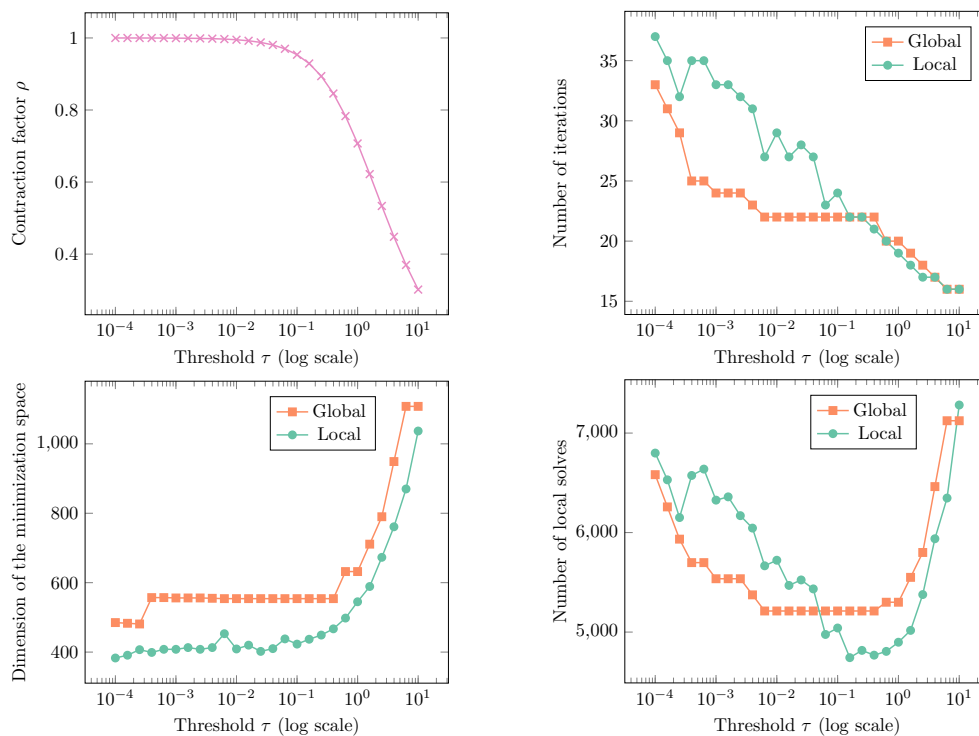
Fig. 4: Influence of the threshold τ on the adaptive MPCG algorithms.

an anonymous referee who stressed the importance of discussing the choice of τ.

REFERENCES

[1] O. Axelsson and I. Kaporin. Error norm estimation and stopping criteria in preconditioned conjugate gradient iterations. *Numer. Linear Algebra Appl.*, 8(4):265–286, 2001.

[2] A. Brandt, J. Brannick, K. Kahl, and I. Livshits. Bootstrap AMG. *SIAM J. Sci. Comput.*, 33(2):612–632, 2011.

[3] C. Brezinski. Multiparameter descent methods. *Linear Algebra Appl.*, 296(1-3):113–141, 1999.

[4] R. Bridson and C. Greif. A multipreconditioned conjugate gradient algorithm. *SIAM J. Matrix Anal. Appl.*, 27(4):1056–1068 (electronic), 2006.

[5] A. Chapman and Y. Saad. Deflated and augmented Krylov subspace techniques. *Numer. Linear Algebra Appl.*, 4(1):43–66, 1997.

[6] Y.-H. De Roeck and P. Le Tallec. Analysis and test of a local domain-decomposition preconditioner. In *Fourth International Symposium on Domain Decomposition Methods for Partial Differential Equations (Moscow, 1990)*, pages 112–128, Philadelphia, PA, 1991. SIAM.

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

[8] V. Dolean, F. Nataf, R. Scheichl, and N. Spillane. Analysis of a two-level Schwarz method with coarse spaces based on local Dirichlet-to-Neumann maps. *Comput. Methods Appl. Math.*, 12(4):391–414, 2012.

[9] Z. Dostál. Conjugate gradient method with preconditioning by projector. *International Journal of Computer Mathematics*, 23(3-4):315–323, 1988.

[10] J. W. Eaton, D. Bateman, and S. Hauberg. *GNU Octave version 3.0.1 manual: a high-level interactive language for numerical computations.* CreateSpace Independent Publishing Platform, 2009. ISBN 1441413006.

[11] Y. Efendiev, J. Galvis, R. Lazarov, and J. Willems. Robust domain decomposition precon-

Variable heterogeneity for $N = 81$ (k-scaling) :

	Global τ–test		Local τ–test		Simultaneous		PPCG		GenEO
E_2/E_1	it	solves	it	solves	it	solves	it	solves	it
1	26	4624	25	4602	14	7212	36	5832	23
10	26	5036	28	5213	14	7212	44	7128	23
10^2	30	6096	25	5164	15	7786	76	12312	21
10^3	23	5374	25	5133	16	8360	126	20412	22
10^4	22	5212	25	5176	16	8360	139	22518	22
10^5	22	5212	24	5041	16	8360	141	22842	23
	$dim < 554$		$dim < 423$		$dim < 1428$		$dim < 353$		$dim < 372$

Variable heterogeneity for $N = 81$ (multiplicity scaling) :

	Global τ–test		Local τ–test		Simultaneous		PPCG		GenEO
E_2/E_1	it	solves	it	solves	it	solves	it	solves	it
1	30	5272	30	5626	20	10656	35	5670	23
10	32	6832	30	5941	21	11230	51	8262	23
10^2	39	9202	34	8276	24	12952	100	16200	23
10^3	34	11688	34	8890	24	12952	204	33048	22
10^4	31	11202	34	8872	25	13526	299	48438	23
10^5	33	11114	35	9089	25	13526	336	54432	23
	$dim < 1365$		$dim < 808$		$dim < 2157$		$dim < 548$		$dim < 662$

Variable number of subdomains for $E_2/E_1 = 10^5$ (k-scaling) :

	Global τ–test		Local τ–test		Simultaneous		PPCG		GenEO
N	it	solves	it	solves	it	solves	it	solves	it
25	20	1784	22	1447	17	2530	69	3450	20
36	24	2392	23	2150	16	3476	87	6264	20
49	20	3364	24	3146	16	4844	110	10780	20
64	21	5264	24	4137	17	7006	152	19456	20
	$dim < 693$		$dim < 379$		$dim < 1193$		$dim < 320$		$dim < 327$

Table 1: Comparison of the five methods for variable heterogeneity and number of Metis subdomains. it: number of iterations. solves: number of local solves.

ditioners for abstract symmetric positive definite bilinear forms. *ESAIM Math. Model. Numer. Anal.*, 46(5):1175–1199, 2012.

[12] C. Farhat and F.-X. Roux. A method of finite element tearing and interconnecting and its parallel solution algorithm. *Int. J. Numer. Meth. Engng.*, 32(6):1205, 1991.

[13] J. Galvis and Y. Efendiev. Domain decomposition preconditioners for multiscale flows in high contrast media: reduced dimension coarse spaces. *Multiscale Model. Simul.*, 8(5):1621–1644, 2010.

[14] A. Gaul, M. H. Gutknecht, J. Liesen, and R. Nabben. A framework for deflated and augmented Krylov subspace methods. *SIAM J. Matrix Anal. Appl.*, 34(2):495–518, 2013.

[15] P. Gosselet, C. Rey, and J. Pebrel. Total and selective reuse of Krylov subspaces for the resolution of sequences of nonlinear structural problems. *Internat. J. Numer. Methods Engrg.*, 94(1):60–83, 2013.

[16] P. Gosselet, D. Rixen, F.-X. Roux, and N. Spillane. Simultaneous FETI and block FETI: Robust domain decomposition with multiple search directions. *Internat. J. Numer. Methods Engrg.*, 104(10):905–927, 2015.

[17] C. Greif, T. Rees, and D. Szyld. Additive Schwarz with variable weights. In *Domain Decomposition Methods in Science and Engineering XXI*. Springer, 2014.

[18] C. Greif, T. Rees, and D. B. Szyld. MPGMRES: a generalized minimum residual method with multiple preconditioners. Technical Report 11-12-23, Temple University, 2014.

[19] R. Haferssas, P. Jolivet, and F. Nataf. A robust coarse space for Optimized Schwarz methods

SORAS-GenEO-2. Technical report, Submitted, hal-01100926, 2015.

[20] P. Havé, R. Masson, F. Nataf, M. Szydlarski, H. Xiang, and T. Zhao. Algebraic domain decomposition methods for highly heterogeneous problems. *SIAM J. Sci. Comput.*, 35(3):C284–C302, 2013.

[21] F. Hecht. *FreeFem++*. Numerical Mathematics and Scientific Computation. Laboratoire J.L. Lions, Université Pierre et Marie Curie, http://www.freefem.org/ff++/, 3.23 edition, 2013.

[22] M. R. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. *J. Research Nat. Bur. Standards*, 49:409–436 (1953), 1952.

[23] W. D. Joubert and T. A. Manteuffel. Iterative methods for nonsymmetric linear systems. *Academic Press, New York*, page 149171, 1990.

[24] S. Kaniel. Estimates for some computational techniques in linear algebra. *Mathematics of Computation*, 20(95):369–378, 1966.

[25] G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM J. Sci. Comput.*, 20(1):359–392 (electronic), 1998.

[26] A. Klawonn, P. Radtke, and O. Rheinbach. Adaptive coarse spaces for BDDC with a transformation of basis. In *Twenty Second International Conference on Domain Decomposition Methods*, 2014.

[27] A. Klawonn, P. Radtke, and O. Rheinbach. FETI-DP methods with an adaptive coarse space. *SIAM J. Numer. Anal.*, 53(1):297–320, 2015.

[28] A. Klawonn and O. Rheinbach. Deflation, projector preconditioning, and balancing in iterative substructuring methods: connections and new results. *SIAM J. Sci. Comput.*, 34(1):A459–A484, 2012.

[29] A. Klawonn and O. Widlund. FETI and Neumann-Neumann iterative substructuring methods: Connections and new results. *Communications on Pure and Applied Mathematics*, 54(1):57–90, 2001.

[30] S. Loisel, H. Nguyen, and R. Scheichl. Optimized Schwarz and 2-Lagrange multiplier methods for multiscale PDEs. Technical report, submitted, 2015.

[31] J. Mandel. Balancing domain decomposition. *Comm. Numer. Methods Engrg.*, 9(3):233–241, 1993.

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

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

[34] G. Meinardus. *Approximation of functions: Theory and numerical methods*. Expanded translation of the German edition. Translated by Larry L. Schumaker. Springer Tracts in Natural Philosophy, Vol. 13. Springer-Verlag New York, Inc., New York, 1967.

[35] F. Nataf, H. Xiang, V. Dolean, and N. Spillane. A coarse space construction based on local Dirichlet-to-Neumann maps. *SIAM J. Sci. Comput.*, 33(4):1623–1642, 2011.

[36] R. A. Nicolaides. Deflation of conjugate gradients with applications to boundary value problems. *SIAM J. Numer. Anal.*, 24(2):355–365, 1987.

[37] A. A. Nikishin and A. Y. Yeremin. Variable block CG algorithms for solving large sparse symmetric positive definite linear systems on parallel computers. I. General iterative scheme. *SIAM J. Matrix Anal. Appl.*, 16(4):1135–1153, 1995.

[38] D. P. O'Leary. The block conjugate gradient algorithm and related methods. *Linear Algebra Appl.*, 29:293–322, 1980.

[39] C. Pechstein and R. Scheichl. Analysis of FETI methods for multiscale PDEs. Part II: interface variation. *Numer. Math.*, 118(3):485–529, 2011.

[40] D. Rixen. *Substructuring and Dual Methods in Structural Analysis*. PhD thesis, Université de Liège, Collection des Publications de la Faculté des Sciences appliquées, n.175, 1997.

[41] D. J. Rixen and C. Farhat. A simple and efficient extension of a class of substructure based preconditioners to heterogeneous structural mechanics problems. *Int. J. Numer. Meth. Engng.*, 44(4):489–516, 1999.

[42] Y. Saad. *Iterative methods for sparse linear systems*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, second edition, 2003.

[43] Y. Saad, M. Yeung, J. Erhel, and F. Guyomarc'h. A deflated version of the conjugate gradient algorithm. *SIAM J. Sci. Comput.*, 21(5):1909–1926, 2000.

[44] B. Sousedík, J. Šístek, and J. Mandel. Adaptive-Multilevel BDDC and its parallel implementation. *Computing*, 95(12):1087–1119, 2013.

[45] N. Spillane. *Robust domain decomposition methods for symmetric positive definite problems*. PhD thesis, Université Pierre et Marie Curie, Paris, 2014.

[46] N. Spillane, V. Dolean, P. Hauret, F. Nataf, C. Pechstein, and R. Scheichl. Abstract robust

coarse spaces for systems of PDEs via generalized eigenproblems in the overlaps. *Numer. Math.*, 126(4):741–770, 2014.

[47] N. Spillane and D. J. Rixen. Automatic spectral coarse spaces for robust FETI and BDD algorithms. *Int. J. Numer. Meth. Engng.*, 95(11):953–990, 2013.

[48] J. M. Tang, R. Nabben, C. Vuik, and Y. A. Erlangga. Comparison of two-level preconditioners derived from deflation, domain decomposition and multigrid methods. *J. Sci. Comput.*, 39(3):340–370, 2009.

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

[50] A. van der Sluis and H. A. van der Vorst. The rate of convergence of conjugate gradients. *Numer. Math.*, 48(5):543–560, 1986.