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 $Ax = b$ 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 $H$. This is to say that $H Ax = H b$ is solved where the condition number $\lambda_{\text{max}}/\lambda_{\text{min}}$ of $HA$ is much smaller than the one of $A$ (with $\lambda_{\text{max}}$ and $\lambda_{\text{min}}$ 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 $H^s$ ($s = 1, \ldots, N$) or for which the usual preconditioner is a sum $\sum_{s=1}^N H^s$. 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

*Centre de Mathématiques Appliquées, École Polytechnique, Route de Saclay, 91128 Palaiseau, France (nicole.spillane@cmap.polytechnique.fr). Financial support by CONICYT through project Fondecyt 3150090.
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 \( H_s \) 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 \( \sum_{s=1}^N H_s \). The estimate is only practical if a lower bound for the eigenvalues of the preconditioned operator is known \((e.g., \text{for the substructuring Domain Decomposition methods } \lambda_{\text{min}} \geq 1 [31, 29, 49])\). 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 \(\rho\), or the coarse space is augmented with contributions coming from several components \(H_s\) (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 Preconditioned 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:

- \((A1)\) \( A \in \mathbb{R}^{n \times n} \) is a symmetric positive definite matrix,
- \((A2)\) \( H \in \mathbb{R}^{n \times n} \) is a symmetric positive definite matrix,
- \((A3)\) \( U \in \mathbb{R}^{n \times n_0} \) is a full rank matrix with \( n_0 < n \).

Throughout the article, the following problem is considered:

Find \( x^* \in \mathbb{R}^n \) such that \( Ax^* = b \)

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

\[
\Pi := I - U(U^T AU)^{-1}U^T A. 
\]

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 \( Ax=b \) preconditioned by \( H \) and \( \Pi = I - U(U^T AU)^{-1}U^T A \) for initial guess \( x_{00} \)

\[
x_0 = \Pi x_{00} + U(U^T AU)^{-1}U^T b; \quad \text{// Improved initial guess}
\]
\[
r_0 = b - Ax_0;
\]
\[
z_0 = Hr_0;
\]
\[
p_0 = \Pi z_0; \quad \text{// Projected initial search direction}
\]

for \( i = 0, 1, \ldots, \) convergence do

\[
q_i = Ap_i;
\]
\[
\alpha_i = \frac{\langle r_i, z_i \rangle}{\langle q_i, p_i \rangle};
\]
\[
x_{i+1} = x_i + \alpha_i p_i;
\]
\[
r_{i+1} = r_i - \alpha_i q_i;
\]
\[
z_{i+1} = Hr_{i+1};
\]
\[
\beta_i = \frac{\langle z_{i+1}, Ap_i \rangle}{\langle p_i, Ap_i \rangle};
\]
\[
p_{i+1} = \Pi z_{i+1} - \beta_i p_i; \quad \text{// Projected search direction}
\]

end

Return \( x_{i+1} \);

If \( U \) is the empty matrix then \( \Pi = I \) 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 \( x_{00} \) given by the user is improved by computing the exact solution in the space range(\( U \)) (this can be rewritten as a projection): \( x_0 = \Pi x_{00} + U(U^T AU)^{-1}U^T b = x_{00} + (I-\Pi)(x_0 - x_{00}) \). The second difference is a consequence. Indeed, the iterative solver must only compute the remaining part of the solution (\( \Pi x_i \)) so all search directions are projected into range(\( \Pi \)). 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(\( U \)) 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 \( n-n_0 (= n-\text{rank}(U)) \) iterations.
2. \( \|x_i - x_i\|_A = \min \{\|x_i - x\|_A : x \in x_{00} + \text{range}(U) + \text{span}\{p_0, \ldots, p_i-1\}\} \).
3. Search directions are pairwise \( A \)-orthogonal: \( \langle p_i, Ap_j \rangle = 0 \) (\( i \neq j \)).
4. Residuals are pairwise \( H \)-orthogonal: \( \langle r_i, Hr_j \rangle = 0 \) 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 \( x_i \in x_{00} + \text{range}(U) + \text{span}\{p_0, \ldots, p_i-1\} \) it is natural that augmenting the space range(\( U \)) leads to better convergence. On the other hand, if the number \( n_0 \) of columns in \( U \) is too large, then the factorization of \( U^T AU \) in the definition of \( \Pi \) 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]):

\[
\frac{\|x_* - x_i\|_A}{\|x_* - x_0\|_A} \leq 2 \left[ \frac{\sqrt{\lambda_{\text{max}}/\lambda_{\text{min}}} - 1}{\sqrt{\lambda_{\text{max}}/\lambda_{\text{min}}} + 1} \right]^i,
\]

where \(\lambda_{\text{max}}\) and \(\lambda_{\text{min}}\) are bounds for the spectrum of the preconditioned operator \(HA\).

For PPCG the same estimate holds but where \(\lambda_{\text{max}}\) and \(\lambda_{\text{min}}\) are replaced by bounds for the non zero eigenvalues of the projected preconditioned operator \(HA\Pi\) [9] (or equivalently the eigenvalues of \(HA\) restricted to range(\(U\))). Consequently, the ideal strategy for choosing \(U\) is to first compute all isolated eigenvalues of \(HA\) and use the corresponding eigenvectors as a basis for the deflation space range(\(U\)). This way, the spectrum of \(HA\Pi\) is clustered and (2.2) guarantees good convergence. Since it is unrealistic to compute the spectrum of \(HA\), there are proposals for approximating \textit{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 \(U\) is initialized with vectors chosen \textit{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 \(d_i\) the error at iteration \(i\): \(d_i = x_* - x_i\), the authors in [1] prove that

\[
\|d_i\|^2_A = \|d_{i-1}\|^2_A - \alpha_{i-1}^2 \|p_{i-1}\|^2_A, \text{ for all } i = 1, \ldots, n - n_0 - 1.
\]

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 \(x_* = x_0 + \sum_{j=0}^{n-n_0-1} \alpha_j p_j = x_* + \sum_{j=0}^{n-n_0-1} \alpha_j p_j\). Then, the A-conjugacy between search directions (item 3 in section 2.1) implies that \(\|d_i\|^2_A = \|x_* - x_i\|^2_A = \sum_{j=i}^{n-n_0-1} \alpha_j^2 \|p_j\|^2_A\). 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 \textit{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., \(d_i \neq 0\)) then (2.3) can be rewritten as

\[
\frac{\|d_{i-1}\|^2_A}{\|d_i\|^2_A} = 1 + \frac{\alpha_{i-1}^2 \|p_{i-1}\|^2_A}{\|d_i\|^2_A} = 1 + \frac{\alpha_{i-1}^2 \|p_{i-1}\|^2_A}{\|r_i\|^2_H} \frac{\|r_i\|^2_H}{\|d_i\|^2_A} = 1 + \frac{\|r_i\|^2_H}{\|d_i\|^2_A}.
\]

Noting that the last factor is related to a Rayleigh quotient for \(HA\) it follows that

\[
\frac{\|d_i\|^2_A}{\|d_{i-1}\|^2_A} \leq \left( 1 + \lambda_{\text{min}} \frac{\|r_i\|^2_H}{\|d_i\|^2_A} \right)^{-1},
\]

where \(\lambda_{\text{min}}\) is again the smallest eigenvalue of the preconditioned operator \(HA\). Indeed, it holds that \(\|y\|^2_{HA} \geq \lambda_{\text{min}} \|y\|^2_A\) for all \(y \in \mathbb{R}^n\) (see, e.g., [49, Lemma C.1]).
From estimate (2.4) it can be deduced that, if there exists \( \tau > 0 \) such that
\[
\| r_i \|_H^2 \leq \| \alpha_{i-1} p_{i-1} \|_A^2
\]
at every iteration \( i = 1, \ldots, j \), then \( \| d_j \|_A / \| d_0 \|_A \leq (1 + \lambda_{\min} \tau)^{-3/2} \). Conversely, to guarantee that the error decreases at least linearly with a given contraction factor \( \rho \) (i.e., \( \| d_i \| / \| d_{i-1} \| \leq \rho \)), it is sufficient to check that:
\[
\| \alpha_{i-1} p_{i-1} \|_A^2 \geq \tau \quad \text{with} \quad \tau := \frac{1 - \rho^2}{\lambda_{\min} \rho^2}.
\]

In the next section two new algorithms are introduced that aim at guaranteeing 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 \( \| d_{i-1} \|_A^2 \), (2.3) can also be rewritten as
\[
\frac{\| d_i \|_A^2}{\| d_{i-1} \|_A^2} = 1 - \frac{\langle r_{i-1}, z_{i-1} \rangle^2}{\langle Ap_{i-1}, p_{i-1} \rangle^2} \cdot \frac{\langle Ap_{i-1}, p_{i-1} \rangle}{\langle Ad_{i-1}, d_{i-1} \rangle} \leq 1 - \lambda_{\min} \frac{\langle r_{i-1}, z_{i-1} \rangle}{\langle Ap_{i-1}, p_{i-1} \rangle}.
\]

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 MPMGRES (multipreconditioned GMRES) algorithm based on MPMGRES [18].

**3. Main Result: Adaptive MPCG.** Two extra assumptions on the preconditioned system \( H Ax = b \) are made:

- **(A4)** a lower bound \( \lambda_{\min} \) for the spectrum of \( HA \) is known,
- **(A5)** the preconditioner \( H \) is a sum of \( N \) contributions: \( H = \sum_{s=1}^N H^s \), with each \( H^s \) 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 \( H^s \) instead of just one contribution corresponding to \( H \).

**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 \( \tau \)-test. If the \( \tau \)-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 \( \tau \)-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 \( H^s \). 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 \( \tau \): 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 \( \tau \)-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 \( Ax = b \) preconditioned by \( \left( \sum_{s=1}^{N} H^s \right) \) and \( \Pi \) for initial guess \( x_0 \). \( \tau \in \mathbb{R}^+ \): chosen by user.

\[
\begin{align*}
1 & \quad x_0 = \Pi x_{00} + U(U^T A U)^{-1} U^T b; \quad r_0 = b - A x_0; \quad Z_0 = H r_0; \quad P_0 = \Pi Z_0; \\
2 & \quad \text{for } i = 0, 1, \ldots, \text{convergence} \text{ do} \\
3 & \quad \quad Q_i = A P_i; \\
4 & \quad \quad \Delta_i = Q_i^T P_i; \quad \gamma_i = P_i^T r_i; \quad \alpha_i = \Delta_i^+ \gamma_i; \\
5 & \quad \quad x_{i+1} = x_i + \alpha_i \gamma_i; \\
6 & \quad \quad r_{i+1} = r_i - Q_i \alpha_i; \\
7 & \quad \quad t_i = \frac{\gamma_i^T \alpha_i}{r_{i+1}^T H r_{i+1}}; \\
8 & \quad \quad \text{if } t_i < \tau \quad \text{// \( \tau \)-test} \\
9 & \quad \quad \quad Z_{i+1} = [H^1 r_{i+1} | \ldots | H^N r_{i+1}]; \quad \text{// Concatenate the } N \text{ vectors} \\
10 & \quad \quad \quad \text{else} \\
11 & \quad \quad \quad \quad Z_{i+1} = H r_{i+1}; \\
12 & \quad \quad \quad \text{end} \\
13 & \quad \quad \Phi_{i,j} = Q_j^T Z_{i+1}; \quad \beta_{i,j} = \Delta_i^+ \Phi_{i,j} \quad \text{for each } j = 0, \ldots, i; \\
14 & \quad \quad P_{i+1} = \Pi Z_{i+1} - \sum_{j=0}^{i} P_j \beta_{i,j}; \\
15 & \quad \text{end} \\
16 & \quad \text{Return } x_{i+1}; \\
\end{align*}
\]

Each time \( t_i < \tau \), an \( N \times N \) matrix \( \Delta_i = P_i^T A P_i \) must be inverted. Since \( P_i \) is the concatenation of contributions from the components \( H^s \) in the preconditioner, it is reasonable to expect that \( \Delta_i \) be full-rank. If it is not, then \( \Delta_i \) is only positive semidefinite and pseudo-inversion (denoted by \( \Delta_i^+ \)) is necessary. In any case, both \( \gamma_i \) and the columns in \( \Phi_{i,j} \) are in range(\( P_i^T \)) = range(\( \Delta_i \)) 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 \( \Delta_i \) to replace \( P_i \) by an \( A \)-orthonormal basis \( P_i \) of range(\( P_i \)) 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 (\( \beta_{i,j} = 0 \) 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):

- \( \mathbf{A}, \mathbf{H}, \mathbf{H}^0, \ldots, \mathbf{H}^N \in \mathbb{R}^{n \times n} \),
- \( x_r, r, b \in \mathbb{R}^n \),
- \( Z_{i+1}, P_{i+1}, Q_{i+1} \in \mathbb{R}^{n \times N} \) or \( \mathbb{R}^n \) depending on the iteration,
- \( \Delta_i \in \mathbb{R}^{N \times N} \) or \( \mathbb{R} \) depending on the iteration,
- \( \gamma_i, \alpha_i \in \mathbb{R}^n \) or \( \mathbb{R} \) depending on the iteration number \( i \),
- \( \Phi_{i,j}, \beta_{i,j} \in \mathbb{R}^{n \times n} \) or \( \mathbb{R} \) or \( \mathbb{R}^{1 \times N} \) or \( \mathbb{R} \) depending on the iteration numbers \( i \) and \( j \).

Note that the quantities \( P_i \alpha_i \) and \( \mathbf{AP}_i \alpha_i \) 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. \( \| x_* - x_0 \|_A = \min \{ \| x_* - x \|_A : x \in x_0 + \text{range}(U) + \sum_{j=0}^{i-1} \text{range}(P_j) \} \).
3. Blocks of search directions are pairwise \( \mathbf{A} \)-orthogonal: \( P_j \mathbf{AP}_i = 0 \) (\( i \neq j \)).
4. Residuals are \( \ell_2 \)-orthogonal to previous search directions: \( P_j r_i = 0 \) (\( i > j \)).
5. Residuals are pairwise \( \mathbf{H} \)-orthogonal: \( \langle \mathbf{H}r_j, r_i \rangle = 0 \) (\( i \neq j \)).

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

**Proof by induction of Properties 3 and 4:**

The case \( i = 1 \) is simple: \( \mathbf{P}_0 \mathbf{AP}_1 = \mathbf{P}_0 \mathbf{AP}_1 \mathbf{A} \mathbf{P}_1 - \mathbf{P}_0 \mathbf{AP}_0 \beta_{0,0} = \Phi_{0,0} - \Delta_0 \mathbf{A} \mathbf{AP}_0 \beta_{0,0} = 0, \) and \( \mathbf{P}_0 \mathbf{r}_1 = \mathbf{P}_0 \mathbf{r}_0 - \mathbf{P}_0 \mathbf{Q}_0 \alpha_0 = \mathbf{0} \).

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

\[
\mathbf{P}_j \mathbf{AP}_{i+1} = \mathbf{P}_j \mathbf{AP}_{i+1} \mathbf{A} \mathbf{P}_i - \sum_{k=0}^{i} \mathbf{P}_j \mathbf{AP}_k \beta_{k,k} = \Phi_{i+1} - \Delta_j \mathbf{A} \mathbf{AP}_i \beta_{j,j} = 0,
\]

\[
\mathbf{P}_j \mathbf{r}_{i+1} = \mathbf{P}_j \mathbf{r}_i - \mathbf{P}_j \mathbf{Q}_i \alpha_i = \begin{cases} 0 & \text{if } j \neq i \text{ since } \mathbf{P}_j \mathbf{r}_i = 0 \text{ and } \mathbf{P}_j \mathbf{Q}_i = 0, \\ \mathbf{P}_i \mathbf{r}_i - \mathbf{P}_i \mathbf{Q}_i \alpha_i = \gamma_i - \Delta_i \mathbf{A} \mathbf{AP}_i \beta_{i,i} = 0 & \text{if } j = i. \end{cases}
\]

**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 \( \langle \mathbf{H} \mathbf{r}_j, \mathbf{r}_i \rangle = \langle \mathbf{H} \mathbf{H} \mathbf{r}_j, \mathbf{r}_i \rangle \) and Property 4 since \( \mathbf{H} \mathbf{H} \mathbf{r}_j \in \sum_{k=0}^{j} \text{range}(\mathbf{P}_k) \).

**Proof of Property 2:**

The minimization result is equivalent to the fact that \( x_* - x_0 \) is the \( \mathbf{A} \)-orthogonal projection of \( x_* - x_0 \) onto \( \text{range}(\mathbf{U}) + \sum_{j=0}^{i-1} \text{range}(\mathbf{P}_j) \). With this, the proof comes down to the \( \mathbf{A} \)-orthogonality between this space and \( x_* - x_i = (x_* - x_0) - (x_i - x_0) \).

For the space \( \text{range}(\mathbf{U}) \) it holds that:

\[
\mathbf{U}^\top \mathbf{A} (x_* - x_i) = \mathbf{U}^\top \mathbf{r}_i = \mathbf{U}^\top \mathbf{H} \mathbf{r}_i = \langle \mathbf{H} \mathbf{r}_i, \mathbf{r}_i \rangle = 0 \text{ since Ker}(\mathbf{H}) = \text{range}(\mathbf{U}).
\]

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

**Proof of Property 1:**

The fact that \( x_{n+1} = x_* \) follows from the observation that \( \text{rank}(\mathbf{P}_i) \geq 1 \) at every iteration until convergence is achieved. This is another way of saying that the
algorithm does not break down. Indeed, assume that rank \((P_i) = 0\) then \(ΠHr_i \in \text{span}(P_0, \ldots, P_{i-1}) = \text{span}(ΠHr_0, \ldots, ΠHr_{i-1})\). Equivalently

\[
Hr_i \in \text{span}(Hr_0, \ldots, Hr_{i-1}) + \text{Ker}(Π) \quad \iff \quad H^{1/2}r_i \in \text{span}(H^{1/2}r_0, \ldots, H^{1/2}r_{i-1}) + H^{-1/2}\text{Ker}(Π).
\]

By Property 5 it holds that \((r_i, Hr_j) = 0\) for \(j = 0, \ldots, i - 1\) and \(r_i \in \text{range}(Π^T)\) so \(r_i \perp \text{Ker}(Π)\). It follows that if rank \((P_i) = 0\), 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 \(x_{n-n_0} = x_*\). □

### 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 \(t_{i-1} \geq τ\) then the relative error is bounded by

\[
\frac{∥x_* - x_i∥_A}{∥x_* - x_{i-1}∥_A} \leq \left(\frac{1}{1 + λ_{\text{min}}τ}\right)^{1/2}.
\]

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

**Proof.** The proof follows the same lines of reasoning as the results in section 2. Recall the notation \(d_i = x_* - x_i\) 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 \(x_I = x_*\) so \(x_* = x_0 + \sum_{i=0}^{I-1} P_iα_i = x_i + \sum_{j=i}^{I-1} P_jα_j\), or equivalently \(d_i = \sum_{j=i}^{I-1} P_jα_j\). The blocks of search directions are pairwise \(A\)-orthogonal (Property 3 in Theorem 3.1) so by subtraction \(∥d_i∥_A^2 = ∥d_{i-1}∥_A^2 - ∥P_{i-1}α_{i-1}∥_A^2\).

Then, recalling that \(∥r_i∥_H^2 = ∥d_i∥_A^2 ≤ λ_{\text{min}}∥d_i∥_A^2\) (by definition of \(λ_{\text{min}}\)), it holds that:

\[
\frac{∥d_{i-1}∥_A^2}{∥d_i∥_A^2} = 1 + \frac{∥P_{i-1}α_{i-1}∥_A^2}{∥r_i∥_H^2} \geq 1 + λ_{\text{min}} \frac{∥P_{i-1}α_{i-1}∥_A^2}{∥r_i∥_H^2}.
\]

The fraction corresponds to the quantity that is measured by the \(τ\)-test. Indeed

\[
(3.1) \quad ∥P_{i-1}α_{i-1}∥_A^2 = ⟨γ_{i-1}, Δ_{i-1}^\top Δ_{i-1} Δ_{i-1}^\top γ_{i-1}⟩ = γ_{i-1}^\top α_{i-1},
\]

so the assumption that \(t_{i-1} \geq τ\) can be rewritten as \(∥P_{i-1}α_{i-1}∥_A^2 ≥ τ∥r_i∥_H^2\) and the result follows. □

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 \(0 < ρ < 1\) and \(τ\) be chosen as \(τ = \frac{1-ρ^2}{2λ_{\text{min}}ρ^2}\). Under the assumptions of Theorem 3.2 it holds that

\[
\frac{∥x_* - x_i∥_A}{∥x_* - x_{i-1}∥_A} ≤ ρ.
\]

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 \(HA\) are smaller than \(1/τ\) then the result of the \(τ\)-test is \(t_i ≥ τ\) at each iteration and Algorithm 2 performs the usual PPCG iterations.
Proof. Begin with $r_i = r_{i-1} - Q_{i-1} \alpha_{i-1}$ and take the inner product by $H_r$:
\[
(H_r, r_i) = -(H_r, A P_{i-1} \alpha_{i-1}) \quad \text{(by Property 5 in Theorem 3.1)}.
\]

An application of the Cauchy-Schwarz inequality in the $A$-norm gives
\[
(H_{r_i}, r_i) \leq (H_{r_i}, A H_{r_i})^{1/2} (P_{i-1} \alpha_{i-1}, A P_{i-1} \alpha_{i-1})^{1/2},
\]
or equivalently
\[
\frac{(H_{r_i}, r_i)}{(H_{r_i}, A H_{r_i})} \leq \frac{(P_{i-1} \alpha_{i-1}, A P_{i-1} \alpha_{i-1})}{(H_{r_i}, r_i)}.
\]

By assumption all eigenvalues of $HA$ are smaller than $1/\tau$ so $\frac{(H_{r_i}, r_i)}{(H_{r_i}, A H_{r_i})} \geq \tau$ and by this, (3.2) and (3.1), the $\tau$-test returns $t_i \geq \tau$. [\[]

**Corollary 3.5.** If the $\tau$-test returns $t_{i-1} < \tau$ then $\|H_{r_i}\|_{H_{r_i}}^2 < \tau \|H_r\|_A^2$. This implies that there is at least one eigenvalue of $HA$ larger than $1/\tau$. Moreover, it holds that $(H_{r_i}, A v) \neq 0$ where $v$ is an eigenvector corresponding to that eigenvalue. This explains why it makes sense to augment the minimization space with the components of $H_{r_i}$.

*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 $A$ is a sum of $N$ contributions: $A = \sum_{s=1}^N A^s$, with each $A^s$ 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 $\Delta_i$ becomes excessive. In this case the *global* $\tau$–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 $\tau$, the cost of factorizing $\Delta_i$, and the reorthogonalization also apply here. Additionally, if at a given iteration $H^t r_{i+1} = 0$, then $t_i^\tau$ is not defined. It cannot occur that $H^t r_{i+1} = 0$ for all values of $s$ unless convergence is achieved ($r_{i+1} = 0$).

If the local $\tau$-tests return $t_i^\tau \leq \tau$ for every $s = 1, \ldots, N$ at a given iteration, then $Z_{i+1} = [H_{r_{i+1}} H_1 r_{i+1} \ldots H_N r_{i+1}]$ 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 $\tau$–test, the evaluation of the local $\tau$–tests relies on quantities that are available with little extra computational work. Indeed $Q_i = \sum_{s=1}^N A^s P_i$, so $A^s P_i$ is available and a simple linear combination of its columns with the coefficients given by $\alpha_i$ is performed, followed by the inner product with $P_i \alpha_i$. It makes sense to look for additional search directions locally if the preconditioner is constructed as a sum of approximate inverses $H^s$ of the components $A^s$ in the operator. This is illustrated with an example of restructuring (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 $\tau$–tests return $t_i^\tau \geq \tau$ 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 $\langle P_i\alpha_i, A^* P_i\alpha_i \rangle \geq \tau \langle r_{i+1}', H'^{-1} r_{i+1}' \rangle$ and summing these estimates over $s = 1, \ldots, N$ gives $\langle P_i\alpha_i, A P_i\alpha_i \rangle \geq \tau \langle H r_{i+1}, r_{i+1} \rangle$. \hfill $\square$

Algorithm 3: Adaptive multipreconditioned CG for $\left( \sum_{s=1}^N A^s \right) x_s = b$ precondioned by $\left( \sum_{s=1}^N H^s \right)$ and $\Pi$ for initial guess $x_{00}$. $\tau \in \mathbb{R}^+$: chosen by user.

```
1  $x_0 = \Pi x_{00} + U(U^T A U)^{-1} U^T b$; $r_0 = b - A x_0$; $Z_0 = H r_0$; $P_0 = \Pi Z_0$;
2  for $i = 0, 1, \ldots$, convergence do
3      $Q_i = A P_i$;
4      $\Delta_i = Q_i^\top P_i$; $\gamma_i = P_i^\top r_i$; $\alpha_i = \Delta_i \gamma_i$;
5      $x_{i+1} = x_i + P_i \alpha_i$;
6      $r_{i+1} = r_i - Q_i \alpha_i$; // (Locally) Adaptive Multipreconditioning
7      $Z_{i+1} = H r_{i+1}$; // initialize $Z_{i+1}$
8      for $s = 1, \ldots, N$ do
9          $t_i^s = \langle P_i \alpha_i, A^s P_i \alpha_i \rangle$;
10         $r_{i+1} = H r_{i+1}$
11         if $t_i^s < \tau$ then // local $\tau$-test
12            $Z_{i+1} = [Z_{i+1} | H^s r_{i+1}]$; // concatenate $Z_{i+1}$ and $H^s r_{i+1}$
13          end
14      end
15      $\Phi_{i,j} = Q_j^\top Z_{i+1}$; $\beta_{i,j} = \Delta_j^\top \Phi_{i,j}$ for each $j = 0, \ldots, i$;
16      $P_{i+1} = \Pi Z_{i+1} - \sum_{j=0}^i P_j \beta_{i,j}$;
17  end
18  Return $x_{i+1}$;
```

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

$$\Omega = \bigcup_{s=1,\ldots,N} \Omega^s; \quad \Omega^s \cap \Omega^t = \emptyset \text{ for all } s \neq t; \quad \Gamma = \bigcup_{s=1,\ldots,N} \partial \Omega^s \setminus \partial \Omega.$$
The original linear system admits the following block form
\[
\begin{pmatrix}
K_{\Gamma\Gamma} & K_{\Gamma I} \\
K_{I\Gamma} & K_{II}
\end{pmatrix}
\begin{pmatrix}
u_I \\
u_I
\end{pmatrix}
= \begin{pmatrix}
f_I \\
f_I
\end{pmatrix}
\implies \begin{cases}
K_{\Gamma\Gamma}u_{\Gamma} + K_{\Gamma I}u_{I} = f_{\Gamma} \\
K_{I\Gamma}u_{\Gamma} + K_{II}u_{I} = f_{I}
\end{cases},
\]
where the subscript \(s\) denotes the restriction to the set of degrees of freedom on \(\Gamma\) and \(\ast\) to the remaining unknowns. From the second line it can be deduced that \(u_{I} = K_{II}^{-1}(f_{I} - K_{II}u_{\Gamma})\) and by injecting this into the first line the problem is reduced to one on the interfaces between subdomains: Find \(u_{\Gamma} \in \mathbb{R}^{n} \ (n := \#(\Gamma)) \) such that
\[
(4.1) \quad Au_{\Gamma} = b, \quad \text{where } A := K_{\Gamma\Gamma} - K_{\Gamma I}K_{II}^{-1}K_{I\Gamma}, \quad \text{and } b := f_{\Gamma} - K_{II}K_{II}^{-1}f_{I}.
\]
This is the linear system that is solved in the BDD method. The result is an approximation \(u_{\Gamma}\) of the solution on \(\Gamma\) and the remaining part of the solution is computed as \(u_{I} = K_{II}^{-1}(f_{I} - K_{II}u_{\Gamma})\). 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 \(K^{s}\) be the local matrices corresponding to the discretization of the same PDE but restricted to each subdomain \(\Omega^{s}\) and write them in block formulation as
\[
K^{s} = \begin{pmatrix}
K_{\Gamma\Gamma}^{s} & K_{\Gamma I}^{s} \\
K_{I\Gamma}^{s} & K_{II}^{s}
\end{pmatrix}
\text{ where } \begin{cases}
\ast_{\Gamma}: \text{degrees of freedom on } \Gamma \cap \partial \Omega^{s}, \\
\ast_{I}: \text{remaining degrees of freedom in } \partial \Omega^{s}.
\end{cases}
\]
Then, define the local Schur complements \(S^{s} := K_{\Gamma\Gamma}^{s} - K_{\Gamma I}^{s}K_{II}^{s}^{-1}K_{I\Gamma}^{s}\). Finally, these can be assembled into the BDD operator already defined in (4.1). Indeed it holds that
\[
(4.2) \quad A = \sum_{s=1}^{N} A^{s}, \quad \text{where for all } s = 1, \ldots, N: \quad A^{s} := R^{s\top}S^{s}R^{s}
\]
and \(R^{s} \in \mathbb{R}^{\#(\Gamma^{s}) \times n}\) is the boolean matrix that, given a vector in \(\mathbb{R}^{n}\), selects the entries in \(\Gamma^{s}\). The fact that \(A\) is a sum of local contributions has now been made apparent and the preconditioner exploits this since it is
\[
(4.3) \quad H := \sum_{s=1}^{N} H^{s}, \quad \text{where for all } s = 1, \ldots, N: \quad H^{s} := R^{s\top}D^{s}R^{s}
\]
and \(\{D^{s}\}_{s=1,...,N}\) is a family of positive definite diagonal matrices that form a partition of unity \((i.e., \text{ they satisfy } \sum_{s=1}^{N} R^{s\top}D^{s}R^{s} = I)\). Once more, \(\dagger\) denotes a pseudo inverse. This last piece of notation reveals a difficulty inherent to BDD: if the local problems \(K^{s}\) are not positive definite then neither are the Schur complements \(S^{s}\). 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:
\[
(4.4) \quad \text{range}(U) = \sum_{s=1}^{N} R^{s\top}D^{s}\text{Ker}(S^{s}).
\]
This offers the double advantage of making all applications of \(\Pi H\) and \(H\Pi^{\top}\) 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 $S^s$: 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 $K^s$ is the matrix associated with the discretization of the variational form $(u,v) \mapsto \int_\Omega \nabla u \cdot \nabla v$. It is obvious that all constant functions are in its kernel. This in turn induces the singularity of the local matrices $K^s$ and of their Schur complements $S^s$.

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: $U$ is a full rank matrix, $A$ and $H$ are symmetric positive definite matrices [31] assembled as the sum of $N$ symmetric positive semidefinite matrices and all eigenvalues of the preconditioned operator $HA$ are larger than $\lambda_{\min} = 1$ [31, 49]. As a result, the two new adaptive MPCG algorithms (namely, Algorithm 2 with the global $\tau$–test, and Algorithm 3 with the local $\tau$–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 $A$ and the preconditioner $H$ (specifically, the Dirichlet solves $K^s_{f,f}^{-1}$ in $S^s$ and the Neumann solves $K^s_{\ast\ast}$ used to compute an application of $S^s_{\ast\ast}$). 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 $A$ are needed if the original formulation of the algorithms is implemented ($P_i$ 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 $A$ 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 $Q_i = A Z_i - \sum_{j=0}^{i-1} Q_j \beta_{i-1,j}$ instead of $Q_i = A P_i$. The connection is that $Q_i = \Pi Q_i$. The values of $\Delta_i$ and all the $\Phi_{i,j}$ remain the same. One more application of the projection $\Pi^\top$ 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 $A$.

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

\begin{equation}
Q^s_i = A^s \left( \Pi Z_i - \sum_{j=0}^{i-1} P_j \beta_{i-1,j} \right) = A^s Z_i - A^s U (U^\top A U)^{-1} (A U)^\top Z_i - \sum_{j=0}^{i-1} Q_j^s \beta_{i-1,j}.
\end{equation}
Again, this allows for a significant reduction in the number of local solves that are incurred by applying $A$ each time some local contributions are selected (when the $\tau$–test returns $t_i^* < \tau$). 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 $Z_i$ 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 $\tilde{P}_i$ is computed instead of $P_i = \Pi Z_i$, with the consequence that in line 6 the projection $\Pi$ is applied to a vector.

Algorithm 4: Algorithm 2 for BDD: Adaptive MPCG with global $\tau$–test

1. $x_0 = \Pi x_0^0 + U(U^\top A U)^{-1}U^\top b$; $r_0 = b - Ax_0$; $Z_0 = H r_0$; $P_0 = \Pi Z_0$;
2. for $i = 0, 1, \ldots$, convergence do
3. $Q_i = A Z_i - \sum_{j=0}^{i-1} Q_j \beta_{i-1,j}$;
4. $\Delta_i = Q_i^\top P_i$; $\gamma_i = P_i^\top r_i$; $\alpha_i = \Delta_i^\top \gamma_i$;
5. $x_{i+1} = x_i + P_i \alpha_i$;
6. $r_{i+1} = \Pi^\top (r_i - Q_i \alpha_i)$;
7. if $t_i < \tau$ then \hspace{1cm} // global $\tau$–test
8. $Z_{i+1} = [H^\top r_{i+1} | \ldots | H^N r_{i+1}]$;
9. else
10. $Z_{i+1} = H r_{i+1}$;
11. end
12. $\Phi_{i,j} = Q_j^\top \Pi Z_{i+1}$; $\beta_{i,j} = \Delta_j^\top \Phi_{i,j}$ for each $j = 0, \ldots, i$;
13. $P_{i+1} = \Pi Z_{i+1} - \sum_{j=0}^{i} P_j \beta_{i,j}$;
14. end
15. Return $x_{i+1}$;

4.3. Connection with previous work. As previously mentioned in section 2.2 it makes sense to generate the deflation space $\text{range}(U)$ with the eigenvectors corresponding to isolated eigenvalues of the preconditioned operator $HA$. 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

$$\text{range}(U) = \text{span}\{R^s^T p_k^s; s = 1, \ldots, N \text{ and } \lambda_k^s \leq \tau\},$$

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

$$(D^{-1} S^* D^{-1}) p_k^s = \lambda_k^s (R^s A R^s^T) p_k^s.$$

Algorithm 1 is then applied. It is proven that with this choice of $U$ the largest eigenvalue of the projected preconditioned operator $HA\Pi$ is bounded by $N/\tau$ where
Algorithm 5: Algorithm 3 for BDD: Adaptive MPCG with local $\tau$-test

1. $x_0 = \Pi x_{00} + U(U^T A U)^{-1} U^T b$; $r_0 = b - A x_0$; $Z_0 = H r_0$;
2. $P_0 = Z_0$;
3. for $i = 0, 1, \ldots$, convergence do
4. $Q_i = \sum_{s=1}^{N} Q_i^s$ where
   \[ Q_i^s = A^s Z_i - A^s U (U^T A U)^{-1} (U A)^T Z_i - \sum_{j=0}^{i-1} Q_j^i \beta_{i-1,j}; \]
5. $\Delta_i = Q_i^\top \tilde{P}_i$; $\gamma_i = \tilde{P}_i^\top r_i$; $\alpha_i = \Delta_i \gamma_i$;
6. $x_{i+1} = x_i + \Pi P_i \alpha_i$;
7. $r_{i+1} = r_i - Q_i \alpha_i$;
8. $Z_{i+1} = H r_{i+1}$;
9. for $s = 1, \ldots, N$ do
10. $t_s^i = \frac{\langle \Pi P_s \alpha_i, Q_s^i \alpha_i \rangle}{r_{i+1}^\top H r_{i+1}}$;
11. if $t_s^i < \tau$ then // local $\tau$-test
12. $Z_{i+1} = [Z_{i+1} | H^s r_{i+1}]$;
13. Subtract $H^s r_{i+1}$ from the first column in $Z_{i+1}$;
14. end
15. end
16. $\Phi_{i,j} = Q_j^\top Z_{i+1}$; $\beta_{i,j} = \Delta_j \Phi_{i,j}$ for each $j = 0, \ldots, i$;
17. $\tilde{P}_{i+1} = Z_{i+1} - \sum_{j=0}^{i} \tilde{P}_j \beta_{i,j}$;
18. end
19. Return $x_{i+1}$;

$N$ is the maximal number of neighbours of a subdomain (including itself). Since $\lambda_{\text{min}}$ is still 1, the classical estimate (2.2) yields $\frac{\|x_* - x_i\|_A}{\|x_* - x_0\|_A} \leq 2 \left[ \frac{\sqrt{N/\tau-1}}{\sqrt{N/\tau+1}} \right]^i$. This result indicates that with local contributions it is possible to guarantee any targeted convergence rate by adjusting $\tau$. 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 $\tau$–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 $\tau$–test returns $t_i \geq \tau$ (respectively, $t_s^i \geq \tau$ 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
\[
\begin{align*}
-\text{div}(\sigma(\mathbf{u})) &= (0, 10)^\top, & \text{in } \Omega, \\
\mathbf{u} &= (0, 0)^\top, & \text{on } \{(x, y) \in \partial \Omega : x = 0\}, \\
\sigma(\mathbf{u}) \cdot \mathbf{n} &= 0, & \text{on the rest of } \partial \Omega \text{ (with } \mathbf{n} \text{ denoting the outward normal).}
\end{align*}
\]
The stress tensor \(\sigma(\mathbf{u})\) is defined by \(\sigma_{ij}(\mathbf{u}) = 2\mu \varepsilon_{ij}(\mathbf{u}) + \lambda \delta_{ij} \text{div}(\mathbf{u})\) for \(i, j = 1, 2\) where
\(\varepsilon_{ij}(\mathbf{u}) = \frac{1}{2} \left( \frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right)\), \(\delta_{ij}\) is the Kronecker symbol and the Lamé coefficients are functions of Young’s modulus \(E\) and Poisson’s ratio \(\nu\) : \(\mu = \frac{E}{2(1+\nu)}\), \(\lambda = \frac{E\nu}{(1+\nu)(1-2\nu)}\).

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 \(h = 1/(10N)\). 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 \(\tau\)-test), Algorithm 5 (Adaptive MPCG with Local \(\tau\)-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 \(\|\mathbf{x}_i - \mathbf{x}_s\|_A\) be smaller than \(10^{-6}\|\mathbf{x}_s\|_A\). 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 \(\tau\) 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 \(\|\mathbf{x}_i - \mathbf{x}_s\|_A/\|\mathbf{x}_s\|_A\), the dimension of the minimization space (see Property 2 in Theorem 3.1), and the number of local solves (number of applications of \(\mathbf{S}^\dagger\) or \(\mathbf{S}^*\)). 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).
Nicole SPILLANE

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

Fig. 1: Geometry of the test problem. There are 19,800 degrees of freedom in the original problem and 3,056 (respectively, 3,458) 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 $\tau$–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 8,586 to 4,302 and 4,176. The algorithm with the global $\tau$–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 $n = 3,458$.

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
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** $\tau$. The threshold $\tau$ plays a very important role in the adaptive MPCG algorithms. The global ($t_i < \tau$) or local ($t_s^i < \tau$) $\tau$-tests determine whether to perform an iteration with one or several search directions, respectively. In Corollary 3.3, it is shown that $\tau$ can be related to a targeted contraction factor $\rho$ 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 $\tau$ ranging between $10^{-4}$ and 10. In Figure 4, the contraction factor $\rho$ 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 $\tau$). For the method with the global $\tau$-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 $\tau$. In fact, for all values of $\tau$ 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 $\tau$ 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 $\tau$ is not unique.

For the local version of the algorithm, the minimal number of local solves is 4743 instead of 5212 with the global $\tau$-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 $\tau$ but also more sensitive to the choice of $\tau$.

**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
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.

Acknowledgements. The author would like to thank P. Gosselet, F. Nataf, D.J. Rixen and F.-X. Roux for some enlightening discussions on this topic as well as
an anonymous referee who stressed the importance of discussing the choice of $\tau$.

REFERENCES

[11] Y. Efendiev, J. Galvis, R. Lazarov, and J. Willems. Robust domain decomposition precon-
An Adaptive MultiPreconditioned Conjugate Gradient Algorithm

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

<table>
<thead>
<tr>
<th>$E_2/E_1$</th>
<th>Global $\tau$-test it solves</th>
<th>Local $\tau$-test it solves</th>
<th>Simultaneous it solves</th>
<th>PPCG it solves</th>
<th>GenEO it solves</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>26 4624</td>
<td>25 4602</td>
<td>14 7212</td>
<td>36 5832</td>
<td>23</td>
</tr>
<tr>
<td>10</td>
<td>26 5036</td>
<td>28 5213</td>
<td>14 7212</td>
<td>44 7128</td>
<td>23</td>
</tr>
<tr>
<td>$10^2$</td>
<td>30 6096</td>
<td>25 5164</td>
<td>15 7786</td>
<td>76 12312</td>
<td>21</td>
</tr>
<tr>
<td>$10^3$</td>
<td>23 5374</td>
<td>25 5133</td>
<td>16 8360</td>
<td>126 20412</td>
<td>22</td>
</tr>
<tr>
<td>$10^4$</td>
<td>22 5212</td>
<td>25 5176</td>
<td>16 8360</td>
<td>139 22518</td>
<td>22</td>
</tr>
<tr>
<td>$10^5$</td>
<td>22 5212</td>
<td>24 5041</td>
<td>16 8360</td>
<td>141 22842</td>
<td>23</td>
</tr>
</tbody>
</table>

$\dim < 554$  $\dim < 423$  $\dim < 1428$  $\dim < 353$  $\dim < 372$

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

<table>
<thead>
<tr>
<th>$E_2/E_1$</th>
<th>Global $\tau$-test it solves</th>
<th>Local $\tau$-test it solves</th>
<th>Simultaneous it solves</th>
<th>PPCG it solves</th>
<th>GenEO it solves</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>30 5272</td>
<td>30 5626</td>
<td>20 10656</td>
<td>35 5670</td>
<td>23</td>
</tr>
<tr>
<td>10</td>
<td>32 6832</td>
<td>30 5941</td>
<td>21 11230</td>
<td>51 8262</td>
<td>23</td>
</tr>
<tr>
<td>$10^2$</td>
<td>39 9202</td>
<td>34 8276</td>
<td>24 12952</td>
<td>100 16200</td>
<td>23</td>
</tr>
<tr>
<td>$10^3$</td>
<td>34 11688</td>
<td>34 8890</td>
<td>24 12952</td>
<td>204 33048</td>
<td>22</td>
</tr>
<tr>
<td>$10^4$</td>
<td>31 11202</td>
<td>34 8872</td>
<td>25 13526</td>
<td>299 48438</td>
<td>23</td>
</tr>
<tr>
<td>$10^5$</td>
<td>33 11114</td>
<td>35 9089</td>
<td>25 13526</td>
<td>336 54432</td>
<td>23</td>
</tr>
</tbody>
</table>

$\dim < 1365$  $\dim < 808$  $\dim < 2157$  $\dim < 548$  $\dim < 662$

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

<table>
<thead>
<tr>
<th>$N$</th>
<th>Global $\tau$-test it solves</th>
<th>Local $\tau$-test it solves</th>
<th>Simultaneous it solves</th>
<th>PPCG it solves</th>
<th>GenEO it solves</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>20 1784</td>
<td>22 1447</td>
<td>17 2530</td>
<td>69 3450</td>
<td>20</td>
</tr>
<tr>
<td>36</td>
<td>24 2392</td>
<td>23 2150</td>
<td>16 3476</td>
<td>87 6264</td>
<td>20</td>
</tr>
<tr>
<td>49</td>
<td>20 3364</td>
<td>24 3146</td>
<td>16 4844</td>
<td>110 10780</td>
<td>20</td>
</tr>
<tr>
<td>64</td>
<td>21 5264</td>
<td>24 4137</td>
<td>17 7006</td>
<td>152 19456</td>
<td>20</td>
</tr>
</tbody>
</table>

$\dim < 693$  $\dim < 379$  $\dim < 1193$  $\dim < 320$  $\dim < 327$

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

---


