Adaptive Multi Preconditioned Conjugate Gradient: Algorithm, Theory and an Application to Domain Decomposition

Nicole Spillane

Adaptive Multi Preconditioned Conjugate Gradient: Algorithm, Theory and an Application to Domain Decomposition

Nicole Spillane *

June 30, 2015

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 Multi Preconditioned CG. We first present this in an abstract framework with the one strong assumption being that a bound for the smallest eigenvalue of the preconditioned operator is available. Then, we apply the algorithm to the Balancing Domain Decomposition method and illustrate its behaviour numerically. In particular we observe that it is 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.

Keywords: Krylov methods, Preconditioners, Conjugate Gradient, Domain Decomposition, Robustness, BDD.

1 Introduction

We consider the problem of solving a symmetric positive definite linear system $\mathbf{A}\mathbf{x}_* = \mathbf{b}$ with the Conjugate Gradient (CG) algorithm [19]. Since we consider possibly ill conditioned systems, a very standard way to accelerate convergence is to use a preconditioner $\mathbf{H}$. This is to say that we solve $\mathbf{H}\mathbf{A}\mathbf{x}_* = \mathbf{H}\mathbf{b}$ where the effective condition number $\lambda_{\max}/\lambda_{\min}$ of $\mathbf{H}\mathbf{A}$ is much smaller than the one of $\mathbf{A}$ ($\lambda_{\max}$ and $\lambda_{\min}$ denote respectively the largest and smallest eigenvalue for which a corresponding eigenvector is a component of the initial error). The reason why this improves the convergence of the iterative solver is that the relative error at a given iteration can be bounded with respect to the effective condition number [31, 21, 39].

One particular type of preconditioning is projection preconditioning [9]. It is also known as deflation [33, 40], augmentation [5] and balancing [20]. A review of these methods with an application to substructuring Domain Decomposition can be found in [25]. 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 on the remainder of the error. If the deflation space is well chosen the effective condition number, and hence the convergence, are greatly improved. In practice the deflation space is often computed as an approximation of the eigenvectors corresponding

*Center for Mathematical Modeling, Facultad de Ciencias Físicas y Matemáticas, Universidad de Chile, Beauchef 851, Santiago, Chile. (nspillane@dim.uchile.cl). Financial support by CONICYT through project Fondecyt 3150090.

to isolated eigenvalues of the preconditioned operator, for example by recycling information from a previous linear solve [14, 46, 2] (see also section 2.2).

The method that we propose is based on the Multi Preconditioned CG algorithm [4] (MPCG). MPCG is itself related to the block CG algorithm [35, 34] and the multi parameter 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 [30] followed by [41, 44, 23, 24] for the substructuring methods FETI and BDD [12, 26, 28]; [32, 8, 43, 13, 11] for the overlapping Additive Schwarz method [45]; and [27, 17] for the Optimized Schwarz method. In this article we will use the Balancing Domain Decomposition Method as an illustration for our new algorithms. We refer to section 4.3 for a brief summary of the deflation space proposed in [44] (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 [36].

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 our new algorithms is MPCG [4]. MPCG has already been applied to Additive Schwarz [16] and FETI (Simultaneous FETI algorithm in [37, 15]). In both cases good convergence was observed. The drawback of these methods is that they generate very many search directions and the cost of minimizing over these search directions and orthogonalizing future contributions against them may become prohibitive specially if very many processors are used.

Instead, our algorithms consider each contribution arising from the application of one $\mathbf{H}^s$ as a *candidate* [2] to augment the space in which we look for the solution (called the minimization space). A theoretical estimate (5) 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 $\lambda_{\min} \geq 1$ [28, 26, 45]). The idea for the new algorithms was first briefly introduced in [42](section 7.3) in the FETI framework. We prove that, given a targeted contraction factor $0 < \rho < 1$, and at a given iteration, 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 at 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 we introduce some classical results for the Projected Preconditioned CG algorithm (PPCG) and prove the new estimate (5). In section 3 we introduce our two new algorithms in a general framework and prove the corresponding theoretical results. In section 4 we apply the algorithms to BDD. Finally, in section 5, we illustrate their behaviour and compare them to existing methods.

2 Projected Preconditioned Conjugate Gradient (PPCG)

Let n, $n_0 \in \mathbb{N}$ with $n_0 < n$. 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.

Throughout the article, we consider the following problem:

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

for a given right hand side $\mathbf{b} \in \mathbb{R}^n$. The natural iterative solver is the conjugate gradient (CG) algorithm. We choose to accelerate it by a (left) preconditioner that we denote 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 $\mathrm{Ker}(\mathbf{\Pi}) = \mathrm{range}(\mathbf{U})$, or explicitly,

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

2.1 Description and well known results

Algorithm 1 describes the Projected Preconditioned Conjugate Gradient (PPCG) algorithm 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 we recover the usual Preconditioned Conjugate Gradient (PCG) algorithm. 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($\mathbf{U}$) (which rewrites as a projection):

$$\mathbf{x}_0 = \mathbf{\Pi}\mathbf{x}_{00} + \mathbf{U}(\mathbf{U}^\top\mathbf{A}\mathbf{U})^{-1}\mathbf{U}^\top\mathbf{b} = \mathbf{x}_{00} + (\mathbf{I} - \mathbf{\Pi})(\mathbf{x}_* - \mathbf{x}_{00}). \tag{2}$$

The second difference is a consequence of this. 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($\mathbf{\Pi}$). Other names for this process are *deflation*, *balancing* and *augmentation* [9, 33, 40, 5, 20]. In the particular field of domain decomposition the deflation space range($\mathbf{U}$) is referred to as the coarse space (see section 3).

Next we state a list of well known results for PPCG [25, 39]:

1. The exact solution is achieved after at most $n - n_0$ iterations.

2. $\|\mathbf{x}_* - \mathbf{x}_i\|_A = \min\{\|\mathbf{x}_* - \mathbf{x}\|_A;\ \mathbf{x} \in \mathbf{x}_{00} + \text{range}(\mathbf{U}) + \text{span}\{\mathbf{p}_0, \ldots, \mathbf{p}_{i-1}\}\}$.
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 ℓ_2-orthogonal to previous search directions: $\langle \mathbf{r}_i, \mathbf{p}_j \rangle = 0$ $(i > j)$.
5. Residuals are pairwise $\mathbf{H}$-orthogonal: $\langle \mathbf{r}_i, \mathbf{H}\mathbf{r}_j \rangle = 0$ for all $(i \neq j)$.

2.2 The choice of deflation space can significantly accelerate convergence

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 are the vectors that will help accelerate convergence.

One way to estimate the relative error of PCG is to use the following convergence result [31, 21] (see also [39](Theorem 6.29)):

$$\frac{\|\mathbf{x}_* - \mathbf{x}_i\|_\mathbf{A}}{\|\mathbf{x}_* - \mathbf{x}_0\|_\mathbf{A}} \leq 2\left[\frac{\sqrt{\lambda_{\max}/\lambda_{\min}} - 1}{\sqrt{\lambda_{\max}/\lambda_{\min}} + 1}\right]^i, \tag{3}$$

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 $\mathbf{HA\Pi}$. 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 (3) guarantees good convergence. Of course it is unrealistic to compute the spectrum of $\mathbf{HA}$. Instead it has been proposed to approximate *a priori* the isolated eigenvalues. An option, popular in domain decomposition, is to solve auxiliary (less costly) eigenvalue problems [30, 41, 44, 23, 24, 32, 8, 43, 13, 11, 27, 17] (see also section 4.3).

The algorithms that we propose in this article are very closely related to deflation except that we perform the augmentation of $\mathbf{U}$ on the fly and not *a priori*. First we derive an estimate that allows us to do that.

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: $\mathbf{d}_i = \mathbf{x}_* - \mathbf{x}_i$, for all $i = 1, \ldots, n - n_0$, the authors in [1] prove that

$$\|\mathbf{d}_i\|^2_\mathbf{A} = \|\mathbf{d}_{i-1}\|^2_\mathbf{A} - \alpha^2_{i-1}\|\mathbf{p}_{i-1}\|^2_\mathbf{A}. \tag{4}$$

The proof holds in three steps: first, by the finite termination property (item 1 in section 2.1), the exact solution can be written as $\mathbf{x}_* = \mathbf{x}_0 + \sum_{j=0}^{n-1} \alpha_j \mathbf{p}_j = \mathbf{x}_i + \sum_{j=i}^{n-1} \alpha_j \mathbf{p}_j$. Then, the $\mathbf{A}$-conjugacy between search directions (item 3 in section 2.1) implies that $\|\mathbf{d}_i\|^2_\mathbf{A} = \|\mathbf{x}_* - \mathbf{x}_i\|^2_\mathbf{A} = \sum_{j=i}^{n-1} \alpha_j^2 \|\mathbf{p}_j\|^2_\mathbf{A}$. Finally (4) follows easily by subtraction.

The authors use this to derive some *a posteriori* error estimates and stopping criteria. Here, we build on the same starting point to derive two adaptive algorithms with the objective of accelerating convergence when necessary. Let's assume that at iteration i we have not yet found the exact solution (*i.e.* $\mathbf{d}_i \neq \mathbf{0}$) then (4) can be rewritten as

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

The second term in the product can be bounded from below by the smallest eigenvalue $\lambda_{\min}$ of the preconditioned operator $\mathbf{HA}$ ($\|\mathbf{d}_i\|^2_{\mathbf{AHA}} \geq \lambda_{\min}\|\mathbf{d}_i\|^2_{\mathbf{A}}$, see for instance [45](Lemma C.1)) so:

$$\frac{\|\mathbf{d}_i\|^2_{\mathbf{A}}}{\|\mathbf{d}_{i-1}\|^2_{\mathbf{A}}} \leq \left(1 + \lambda_{\min}\frac{\|\alpha_{i-1}\mathbf{p}_{i-1}\|^2_{\mathbf{A}}}{\|\mathbf{r}_i\|^2_{\mathbf{H}}}\right)^{-1}. \tag{5}$$

From estimate (5) we deduce that, if there exists $\tau > 0$ such that $\tau\|\mathbf{r}_i\|^2_{\mathbf{H}} \leq \|\alpha_{i-1}\mathbf{p}_{i-1}\|^2_{\mathbf{A}}$ 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:

$$\frac{\|\alpha_{i-1}\mathbf{p}_{i-1}\|^2_{\mathbf{A}}}{\|\mathbf{r}_i\|^2_{\mathbf{H}}} \geq \tau \text{ with } \tau := \frac{1-\rho^2}{\lambda_{\min}\rho^2}. \tag{6}$$

In the next section we introduce two new algorithms that aim at guarantying a targeted convergence bound. They are based on evaluating at each iteration whether (6) holds or not. In the case where it doesn't we propose to accelerate convergence.

Remark 1. *After division by* $\|\mathbf{d}_{i-1}\|^2_{\mathbf{A}}$, (4) *can also be rewritten as*

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

We mention this estimate because it arises more naturally and we have not seen it before. However we did not use it in our adaptive algorithms.

3 Main Result: New Adaptive Algorithm

We make two extra assumptions on the preconditioned system $\mathbf{HAx}_* = \mathbf{Hb}$:

(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 : $\mathbf{H} = \sum_{s=1}^{N} \mathbf{H}^s$, with each $\mathbf{H}^s$ symmetric and positive semi definite.

The motivation for these two assumptions is directly connected to the two main ingredients in the adaption step of our adaptive algorithm. Indeed Assumption **(A4)** guarantees that the terms in the relative error estimate (5) can be evaluated and consequently that this estimate can be used as an indicator of whether we need to adapt the algorithm or not (*i.e.* accelerate convergence). Assumption **(A5)** is just as vital since, when a lack of robustness is detected, convergence will be improved by searching for the next approximate solution in a 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 algorithm and introduces some new notation. The new algorithm is designed to adapt automatically if convergence is too slow. More precisely, given a threshold $\tau \in \mathbb{R}^+$ chosen by the user, the adaptation step is between lines 7 and 12. We will refer to the test in line 8 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) Multi Preconditioned CG (MPCG) algorithm [4] for the N preconditioners $\mathbf{H}^s$. There are two extreme

Algorithm 2: New algorithm 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 $\quad \mathbf{Q}_i = \mathbf{A}\mathbf{P}_i$;
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 $\quad \mathbf{r}_{i+1} = \mathbf{r}_i - \mathbf{Q}_i\boldsymbol{\alpha}_i$;
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** `// τ-test`
9 $\quad\quad \mathbf{Z}_{i+1} = \left[\mathbf{H}^1\mathbf{r}_{i+1} \,|\, \ldots \,|\, \mathbf{H}^N\mathbf{r}_{i+1}\right]$; `// Concatenate the N vectors`
10 $\quad$ **else**
11 $\quad\quad \mathbf{Z}_{i+1} = \mathbf{H}\mathbf{r}_{i+1}$;
12 $\quad$ **end**
13 $\quad \mathbf{\Phi}_{i,j} = \mathbf{Q}_j^\top\mathbf{Z}_{i+1}$; $\quad \boldsymbol{\beta}_{i,j} = \mathbf{\Delta}_j^\dagger\mathbf{\Phi}_{i,j}$ for each $j = 0, \ldots, i$;
14 $\quad \mathbf{P}_{i+1} = \mathbf{\Pi}\mathbf{Z}_{i+1} - \sum_{j=0}^{i} \mathbf{P}_j\boldsymbol{\beta}_{i,j}$;
15 **end**
16 Return $\mathbf{x}_{i+1}$;

choices for τ: if $\tau = 0$ then we recover the usual PPCG iterations and if $\tau > \lambda_{\max}$ then we recover the projected MPCG algorithm.

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-ranked. If it is not then $\mathbf{\Delta}_i$ is only positive semi definite 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 range$(\mathbf{P}_i^T)$ = range$(\mathbf{\Delta}_i)$ so that the iteration is always well defined. The exact same occurs in the Simultaneous FETI and Block FETI algorithms [15]. 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 range$(\mathbf{P}_i)$ and simplify future orthogonalization steps.

We have presented the algorithm 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 we do not comment further on this for several reasons: *(i)* we plan for our algorithm to solve the problem in few iterations so the cost of reorthogonalization will be low, *(ii)* for substructuring methods, which is the application that we propose in the next section, it has been observed that full reorthogonalization is in fact crucial because of numerical errors.

Remark 2. *For clarity we give the size of the different variables (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,*

- $\boldsymbol{\Delta}_i \in \mathbb{R}^{N\times N}$ *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*,*
- $\boldsymbol{\Phi}_{i,j}, \boldsymbol{\beta}_{i,j} \in \mathbb{R}^{N\times N}$ *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 we only use the notation* $\langle\cdot,\cdot\rangle$ *for computing the inner product between two vectors.*

3.2 The usual PPCG properties hold for Algorithm 2

In Theorem 1 we prove results similar to the ones stated in section 2.1 for PPCG.

Theorem 1. *Algorithm 2 satisfies the five following properties:*

1. *The exact solution is achieved after at most* $n - n_0$ *iterations.*
2. $\|\mathbf{x}_* - \mathbf{x}_i\|_A = \min\left\{\|\mathbf{x}_* - \mathbf{x}\|_A;\ \mathbf{x} \in \mathbf{x}_{00} + \mathit{range}(\mathbf{U}) + \sum_{j=0}^{i-1}\mathrm{range}(\mathbf{P}_j)\right\}$.
3. *Blocs 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 projection $\boldsymbol{\Pi}$. In particular note that $\boldsymbol{\Pi}^\top\mathbf{A} = \mathbf{A}\boldsymbol{\Pi}$; $\boldsymbol{\Pi}\mathbf{P}_i = \mathbf{P}_i$; $\boldsymbol{\Pi}^\top\mathbf{r}_i = \mathbf{r}_i$.
Proof by induction of Properties 3 and 4:
The case $i = 1$ is simple: $\mathbf{P}_0^\top\mathbf{A}\mathbf{P}_1 = \mathbf{P}_0^\top\mathbf{A}\boldsymbol{\Pi}\mathbf{Z}_1 - \mathbf{P}_0^\top\mathbf{A}\mathbf{P}_0\boldsymbol{\beta}_{0,0} = \boldsymbol{\Phi}_{0,0} - \boldsymbol{\Delta}_0\boldsymbol{\Delta}_0^\dagger\boldsymbol{\Phi}_{0,0} = \mathbf{0}$,
and $\mathbf{P}_0^\top\mathbf{r}_1 = \mathbf{P}_0^\top\mathbf{r}_0 - \mathbf{P}_0^\top\mathbf{Q}_0\boldsymbol{\alpha}_0 = \boldsymbol{\gamma}_0 - \boldsymbol{\Delta}_0\boldsymbol{\Delta}_0^\dagger\boldsymbol{\gamma}_0 = \mathbf{0}$.

Next we assume that both properties hold for a given $i \geq 1$ and deduce them for $i+1$. Let $j \leq i$, then

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

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 $\langle\mathbf{H}\mathbf{r}_j, \mathbf{r}_i\rangle = \langle\boldsymbol{\Pi}\mathbf{H}\mathbf{r}_j, \mathbf{r}_i\rangle$ and Property 4 since $\boldsymbol{\Pi}\mathbf{H}\mathbf{r}_j \in \sum_{k=0}^{j}\mathrm{range}\,(\mathbf{P}_k)$.
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 $\mathrm{range}(\mathbf{U}) + \sum_{j=0}^{i-1}\mathrm{range}(\mathbf{P}_j)$. With this, the proof comes down to the $\mathbf{A}$-orthogonality between this space and $\mathbf{x}_* - \mathbf{x}_i = (\mathbf{x}_* - \mathbf{x}_{00}) - (\mathbf{x}_i - \mathbf{x}_{00})$. We begin with the space $\mathrm{range}(\mathbf{U})$:

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

For any of the spaces $\mathrm{range}(\mathbf{P}_j)$ $(j = 0, \ldots, i-1)$ 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 $\mathrm{rank}\,(\mathbf{P}_i) \geq 1$ 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 $\mathbf{\Pi H r}_i \in \text{span}\{\mathbf{P}_0, \dots, \mathbf{P}_{i-1}\} = \text{span}\{\mathbf{\Pi H r}_0, \dots, \mathbf{\Pi H r}_{i-1}\}$. Equivalently we may write

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

By Property 5 it holds that $\langle \mathbf{r}_i, \mathbf{H r}_j \rangle = 0$ for $j = 0, \dots, i-1$ and $\mathbf{r}_i \in \text{Im}(\mathbf{\Pi}^\top)$ 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 $\mathbf{x}_{n-n_0} = \mathbf{x}_*$. □

3.3 Convergence Results

The following theorems hold.

Theorem 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 as for the results in section 2. Once more we use the notation $\mathbf{d}_i = \mathbf{x}_* - \mathbf{x}_i$ for the error at iteration i. By the finite termination property in Theorem 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 blocs of search directions are pairwise $\mathbf{A}$-orthogonal (Property 3 in Theorem 1) so by subtraction we obtain $\|\mathbf{d}_i\|_A^2 = \|\mathbf{d}_{i-1}\|_A^2 - \|\mathbf{P}_{i-1}\boldsymbol{\alpha}_{i-1}\|_A^2$. Then, recalling that $\|\mathbf{r}_i\|_H^2 = \|\mathbf{d}_i\|_{AHA}^2 \geq \lambda_{\min}\|\mathbf{d}_i\|_A^2$ (by definition of $\lambda_{\min}$) it holds that :

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

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

$$\|\mathbf{P}_{i-1}\boldsymbol{\alpha}_{i-1}\|_A^2 = \langle \boldsymbol{\gamma}_{i-1}, \mathbf{\Delta}_{i-1}^\dagger \mathbf{\Delta}_{i-1} \mathbf{\Delta}_{i-1}^\dagger \boldsymbol{\gamma}_{i-1} \rangle = \boldsymbol{\gamma}_{i-1}^\top \boldsymbol{\alpha}_{i-1}, \tag{7}$$

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

The next theorem states that, if the problem is well conditioned, then no adaption 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. *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. We begin with $\mathbf{r}_i = \mathbf{r}_{i-1} - \mathbf{Q}_{i-1}\boldsymbol{\alpha}_{i-1}$ and take the inner product by $\mathbf{H r}_i$:

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

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

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

or equivalently

$$\frac{\langle \mathbf{H}\mathbf{r}_i, \mathbf{r}_i\rangle}{\langle \mathbf{H}\mathbf{r}_i, \mathbf{A}\mathbf{H}\mathbf{r}_i\rangle} \leq \frac{\langle \mathbf{P}_{i-1}\boldsymbol{\alpha}_{i-1}, \mathbf{A}\mathbf{P}_{i-1}\boldsymbol{\alpha}_{i-1}\rangle}{\langle \mathbf{H}\mathbf{r}_i, \mathbf{r}_i\rangle}. \tag{8}$$

By assumption all eigenvalues of $\mathbf{HA}$ are smaller than $1/\tau$ so $\frac{\langle \mathbf{H}\mathbf{r}_i, \mathbf{r}_i\rangle}{\langle \mathbf{H}\mathbf{r}_i, \mathbf{A}\mathbf{H}\mathbf{r}_i\rangle} \geq \tau$ and by this, (8) and (7) the τ–test returns $t_i \geq \tau$. □

Corollary 1. *If the τ–test returns $t_{i-1} < \tau$ then $\|\mathbf{H}\mathbf{r}_i\|^2_{\mathbf{H}^{-1}} < \tau\|\mathbf{H}\mathbf{r}_i\|^2_{\mathbf{A}}$. This implies that there is at least one eigenvalue of $\mathbf{HA}$ that is larger than $1/\tau$. Moreover, it holds that $\langle \mathbf{H}\mathbf{r}_i, \mathbf{A}\mathbf{v}\rangle \neq 0$ 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 $\mathbf{H}\mathbf{r}_i$.*

Proof. The existence of an eigenvalue larger that $1/\tau$ follows easily from (8) and (7). □

3.4 Alternate Algorithm

We make one more assumption:

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

In cases where the number N of components in the preconditioner is very large it may occur that the cost of factorizing $\boldsymbol{\Delta}_i$ become excessive. In this case we propose to swap the *global* τ–test in Algorithm 2 (line 8) for N tests that are *local* and deflate only the *local* components that are problematic. This is presented in Algorithm 3 and the adaptation step is 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. This is not a problem since in this case $\mathbf{H}^s\mathbf{r}_{i+1}$ does not contribute to the preconditioned residual and can be discarded right away. 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 $\mathbf{Z}_{i+1} = \left[\mathbf{H}\mathbf{r}_{i+1} | \mathbf{H}^1\mathbf{r}_{i+1} | \ldots | \mathbf{H}^N\mathbf{r}_{i+1}\right]$ 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 all we need to perform is a linear combination of its columns with the coefficients given by $\boldsymbol{\alpha}_i$ and the inner product by $\mathbf{P}_i\boldsymbol{\alpha}_i$. 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. We illustrate this with the example of substructuring (domain decomposition) solvers in the next section.

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

Theorem 4.
- *The five properties proved in Theorem 1 hold.*
- *The convergence bound in Theorem 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 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: New algorithm for $\left(\sum_{s=1}^{N}\mathbf{A}^s\right)\mathbf{x}_* = \mathbf{b}$ 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.

$\mathbf{x}_0 = \mathbf{\Pi}\mathbf{x}_{00} + \mathbf{U}(\mathbf{U}^\top\mathbf{A}\mathbf{U})^{-1}\mathbf{U}^\top\mathbf{b}$; $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$; $\mathbf{Z}_0 = \mathbf{H}\mathbf{r}_0$; $\mathbf{P}_0 = \mathbf{\Pi}\mathbf{Z}_0$;
for $i = 0, 1, \ldots,$ *convergence* **do**
  $\mathbf{Q}_i = \mathbf{A}\mathbf{P}_i$;
  $\mathbf{\Delta}_i = \mathbf{Q}_i^\top\mathbf{P}_i$; $\quad \boldsymbol{\gamma}_i = \mathbf{P}_i{}^\top\mathbf{r}_i$; $\quad \boldsymbol{\alpha}_i = \mathbf{\Delta}_i^\dagger\boldsymbol{\gamma}_i$;
  $\mathbf{x}_{i+1} = \mathbf{x}_i + \mathbf{P}_i\boldsymbol{\alpha}_i$;
  $\mathbf{r}_{i+1} = \mathbf{r}_i - \mathbf{Q}_i\boldsymbol{\alpha}_i$;
  $\mathbf{Z}_{i+1} = \mathbf{H}\mathbf{r}_{i+1}$; // initialize $\mathbf{Z}_{i+1}$
  for $s = 1, \ldots, N$ **do**
    $t_i^s = \dfrac{\langle \mathbf{P}_i\boldsymbol{\alpha}_i, \mathbf{A}^s\mathbf{P}_i\boldsymbol{\alpha}_i\rangle}{\mathbf{r}_{i+1}^\top\mathbf{H}^s\mathbf{r}_{i+1}}$;
    if $t_i^s < \tau$ **then** // local τ-test
      $\mathbf{Z}_{i+1} = [\mathbf{Z}_{i+1} \,|\, \mathbf{H}^s\mathbf{r}_{i+1}]$; // concatenate $\mathbf{Z}_{i+1}$ and $\mathbf{H}^s\mathbf{r}_{i+1}$
    end
  end
  $\mathbf{\Phi}_{i,j} = \mathbf{Q}_j^\top\mathbf{Z}_{i+1}$; $\quad \boldsymbol{\beta}_{i,j} = \mathbf{\Delta}_j^\dagger\mathbf{\Phi}_{i,j}$ for each $j = 0, \ldots, i$;
  $\mathbf{P}_{i+1} = \mathbf{\Pi}\mathbf{Z}_{i+1} - \sum_{j=0}^{i}\mathbf{P}_j\boldsymbol{\beta}_{i,j}$;
end
Return $\mathbf{x}_{i+1}$;

4 Application: Balancing Domain Decomposition (BDD)

Domain Decomposition methods are linear solvers for parallel computers. 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 we will focus on one of the so called substructuring methods: Balancing Domain Decomposition, or BDD [28].

4.1 Notation and Introduction of the BDD formulation

Let's assume that we are given a linear system $\mathbf{K}\mathbf{u} = \mathbf{f}$ 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 $\mathbb{R}^3$. Let's also 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:

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

Now, the original linear system admits the following block formulation

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

where the subscript $*_\Gamma$ denotes the restriction to the set of degrees of freedom on Γ and $*_I$ to the remainder. From the second line we deduce 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 we reduce the problem to one on the interfaces between subdomains: Find $\mathbf{u}_\Gamma \in \mathbb{R}^n$ ($n := \#(\Gamma)$) such that

$$\mathbf{A}\mathbf{u}_\Gamma = \mathbf{b}, \text{ where } \mathbf{A} := \mathbf{K}_{\Gamma\Gamma} - \mathbf{K}_{\Gamma I}\mathbf{K}_{II}^{-1}\mathbf{K}_{I\Gamma} \text{ and } \mathbf{b} := \mathbf{f}_\Gamma - \mathbf{K}_{\Gamma I}\mathbf{K}_{II}^{-1}\mathbf{f}_I. \tag{9}$$

This is the linear system that is solved for BDD. 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 in terms of parallel computing and fits the framework for our new algorithms we need to rewrite (9) in 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 it in block formulation as

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

Then, define the local Schur complements $\mathbf{S}^s := \mathbf{K}^s_{\Gamma^s\Gamma^s} - \mathbf{K}^s_{\Gamma^s I^s}(\mathbf{K}^s_{I^s I^s})^{-1}\mathbf{K}^s_{I^s\Gamma^s}$. Finally, these can be assembled into the BDD operator already defined in (9). Indeed it holds that

$$\mathbf{A} = \sum_{s=1}^{N} \mathbf{A}^s, \text{ where for all } s = 1, \dots, N: \quad \mathbf{A}^s := {\mathbf{R}^s}^\top \mathbf{S}^s \mathbf{R}^s \tag{10}$$

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

$$\mathbf{H} := \sum_{s=1}^{N} \mathbf{H}^s, \text{ where for all } s = 1, \dots, N: \quad \mathbf{H}^s := {\mathbf{R}^s}^\top \mathbf{D}^s \mathbf{S}^{s\dagger} \mathbf{D}^s \mathbf{R}^s, \tag{11}$$

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, † 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 has been overcome since [28] by adding a deflation step to the Neumann Neumann algorithm [6]. The deflation, or *coarse* space, is chosen as:

$$\text{range}(\mathbf{U}) = \sum_{s=1}^{N} {\mathbf{R}^s}^\top \mathbf{D}^s \text{Ker}(\mathbf{S}^s). \tag{12}$$

This offers the double advantage of making all applications of $\mathbf{\Pi H}$ and $\mathbf{H\Pi}^\top$ 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].

4.2 New Adaptive BDD

There were five assumptions in sections 2 and 3. With the notation introduced in (10), (11) and (12) all five of these assumptions hold: $\mathbf{U}$ is a full rank matrix, $\mathbf{A}$ and $\mathbf{H}$ are symmetric positive definite matrices [28] assembled as the sum of N symmetric positive semi definite matrices and all eigenvalues of the preconditioned operator $\mathbf{HA}$ are larger than $\lambda_{\min} = 1$ [28, 45]. Thanks to this, we can straightforwardly apply our two new algorithms (namely Algorithm 2 with the global τ–test

and Algorithm 3 with the local τ–test) to the BDD linear system (9). Moreover, the theoretical results in Theorems 1, 2, 3, 4 and Corollary 1 hold.

It is well known that the most time and resource consuming operations in a BDD algorithm are the local solves required by any application of $\mathbf{A}$ (Dirichlet solves in $\mathbf{S}^s$) and $\mathbf{H}$ (Neumann solves in $\mathbf{S}^{s\dagger}$). We observe that the cost of preconditioning is the same in one iteration of our new algorithms as in an iteration of PPCG for BDD. However in iterations where we select multiple search directions, 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 we are interested in high performance computing we propose Algorithms 4 and 5 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 [15] (equation (10)) we have 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 section we will compare several methods in terms of numbers of local solves and observe that from that point of view our new solvers are extremely competitive.

In line 3 of Algorithm 4 we propose to compute $\tilde{\mathbf{Q}}_i = \mathbf{A}\mathbf{Z}_i - \sum_{j=0}^{i-1} \tilde{\mathbf{Q}}_j \boldsymbol{\beta}_{i-1,j}$ instead of $\mathbf{Q}_i = \mathbf{A}\mathbf{P}_i$. The connection is that $\mathbf{Q}_i = \boldsymbol{\Pi}\tilde{\mathbf{Q}}_i$. The values of $\boldsymbol{\Delta}_i$ and all the $\boldsymbol{\Phi}_{i,j}$ remain the same. One more application of the projection $\boldsymbol{\Pi}^\top$ to a vector is performed in line 6. We are confident that this is negligible compared to the significant gain incurred by the localization of the applications of $\mathbf{A}$.

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

$$\begin{aligned}\mathbf{Q}_i^s &= \mathbf{A}^s \left(\boldsymbol{\Pi}\mathbf{Z}_i - \sum_{j=0}^{i-1} \mathbf{P}_j \boldsymbol{\beta}_{i-1,j} \right) \\ &= \mathbf{A}^s\mathbf{Z}_i - \mathbf{A}^s\mathbf{U}(\mathbf{U}^\top\mathbf{A}\mathbf{U})^{-1}(\mathbf{A}\mathbf{U})^\top\mathbf{Z}_i - \sum_{j=0}^{i-1} \mathbf{Q}_j^s \boldsymbol{\beta}_{i-1,j}.\end{aligned} \tag{13}$$

Once more, this allows to reduce significantly the number of Dirichlet solves 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 since it consists in some low rank corrections. An additional modification is that in line 13 we make the first column in $\mathbf{Z}_i$ more sparse by subtracting local contributions from the global vector when they contribute independently to the minimization space. This also saves Dirichlet solves. Finally, in order to save some projection steps we compute $\tilde{\mathbf{P}}_i$ instead of $\mathbf{P}_i = \boldsymbol{\Pi}\tilde{\mathbf{P}}_i$ with the consequence that in line 6 the projection $\boldsymbol{\Pi}$ is applied to a vector.

4.3 Why this is expected to give good results – connection with previous work

As already 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 [30, 41, 44, 42, 23] propose instead to approximate this space with local contributions. More precisely in [44] (Theorem 2.11 and Remark 2.9), and given a threshold $\tau \in \mathbb{R}^+$, the coarse space is chosen so that

$$\text{range}(\mathbf{U}) = \text{span}\{{\mathbf{R}^s}^\top \mathbf{p}_k^s; s = 1, \ldots, N \text{ and } \lambda_k^s \leq \tau\} \tag{14}$$

Algorithm 4: Algorithm 2 applied to BDD: Adaptive Algorithm with global τ–test

$\mathbf{x}_0 = \mathbf{\Pi}\mathbf{x}_{00} + \mathbf{U}(\mathbf{U}^\top\mathbf{A}\mathbf{U})^{-1}\mathbf{U}^\top\mathbf{b}$; $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$; $\mathbf{Z}_0 = \mathbf{H}\mathbf{r}_0$; $\mathbf{P}_0 = \mathbf{\Pi}\mathbf{Z}_0$;
for $i = 0, 1, \ldots,$ *convergence* **do**
  $\tilde{\mathbf{Q}}_i = \mathbf{A}\mathbf{Z}_i - \sum_{j=0}^{i-1} \tilde{\mathbf{Q}}_j \boldsymbol{\beta}_{i-1,j}$;
  $\boldsymbol{\Delta}_i = \tilde{\mathbf{Q}}_i^\top \mathbf{P}_i$; $\boldsymbol{\gamma}_i = {\mathbf{P}_i}^\top \mathbf{r}_i$; $\boldsymbol{\alpha}_i = \boldsymbol{\Delta}_i^\dagger \boldsymbol{\gamma}_i$;
  $\mathbf{x}_{i+1} = \mathbf{x}_i + \mathbf{P}_i\boldsymbol{\alpha}_i$;
  $\mathbf{r}_{i+1} = \mathbf{\Pi}^\top(\mathbf{r}_i - (\tilde{\mathbf{Q}}_i\boldsymbol{\alpha}_i))$;
  $t_i = \dfrac{\boldsymbol{\gamma}_i^\top \boldsymbol{\alpha}_i}{\mathbf{r}_{i+1}^\top \mathbf{H}\mathbf{r}_{i+1}}$;
  if $t_i < \tau$ **then** `// global τ-test`
    $\mathbf{Z}_{i+1} = \left[\mathbf{H}^1\mathbf{r}_{i+1} \,|\, \ldots \,|\, \mathbf{H}^N\mathbf{r}_{i+1}\right]$;
  else
    $\mathbf{Z}_{i+1} = \mathbf{H}\mathbf{r}_{i+1}$;
  end
  $\boldsymbol{\Phi}_{i,j} = \tilde{\mathbf{Q}}_j^\top \mathbf{\Pi}\mathbf{Z}_{i+1}$; $\boldsymbol{\beta}_{i,j} = \boldsymbol{\Delta}_j^\dagger \boldsymbol{\Phi}_{i,j}$ for each $j = 0, \ldots, i$;
  $\mathbf{P}_{i+1} = \mathbf{\Pi}\,\mathbf{Z}_{i+1} - \sum_{j=0}^{i} \mathbf{P}_j \boldsymbol{\beta}_{i,j}$;
end
Return $\mathbf{x}_{i+1}$;

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 proved that with this choice of $\mathbf{U}$ the largest eigenvalue of the projected preconditioned operator $\mathbf{HA\Pi}$ is bounded by $\mathcal{N}/\tau$ where $\mathcal{N}$ is the maximal number of neighbours of a subdomain (including itself). Since $\lambda_{\min}$ is still 1, the classical estimate (3) yields $\frac{\|\mathbf{x}_* - \mathbf{x}_i\|_{\mathbf{A}}}{\|\mathbf{x}_* - \mathbf{x}_0\|_{\mathbf{A}}} \leq 2\left[\frac{\sqrt{\mathcal{N}/\tau}-1}{\sqrt{\mathcal{N}/\tau}+1}\right]^i$. This result tells us 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 as good a convergence with a coarse space that is enriched on the fly. Even though the τ–test in our algorithms does not measure exactly the same quantities as the generalized eigenvalue problem we expect to capture the relevant quantities within the iterations of our two algorithms and ensure fast convergence. We will compare both approaches in the next section.

Remark 3. *The theoretical result for our new 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) we guarantee that the error will decrease linearly. What we do not prove however is how often this will happen. This is somewhat similar to the GenEO result since in that case no result is given for the size of the coarse space. In future work we plan to study these questions but this can only be done for some particular choices of partial differential equations, geometries and discretizations.*

Algorithm 5: Algorithm 3 applied to BDD: Adaptive Algorithm with local τ–test

$\mathbf{x}_0 = \mathbf{\Pi}\mathbf{x}_{00} + \mathbf{U}(\mathbf{U}^\top\mathbf{A}\mathbf{U})^{-1}\mathbf{U}^\top\mathbf{b}$; $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$; $\mathbf{Z}_0 = \mathbf{H}\mathbf{r}_0$;
$\tilde{\mathbf{P}}_0 = \mathbf{Z}_0$;
for $i = 0, 1, \ldots,$ *convergence* **do**
  $\mathbf{Q}_i = \sum_{s=1}^{N} \mathbf{Q}_i^s$ where $\mathbf{Q}_i^s = \mathbf{A}^s\mathbf{Z}_i - \mathbf{A}^s\mathbf{U}(\mathbf{U}^\top\mathbf{A}\mathbf{U})^{-1}(\mathbf{A}\mathbf{U})^\top\mathbf{Z}_i - \sum_{j=0}^{i-1} \mathbf{Q}_j^s\boldsymbol{\beta}_{i-1,j}$;
  $\mathbf{\Delta}_i = \mathbf{Q}_i^\top\tilde{\mathbf{P}}_i$; $\boldsymbol{\gamma}_i = \tilde{\mathbf{P}}_i^\top\mathbf{r}_i$; $\boldsymbol{\alpha}_i = \mathbf{\Delta}_i^\dagger\boldsymbol{\gamma}_i$;
  $\mathbf{x}_{i+1} = \mathbf{x}_i + \mathbf{\Pi}\tilde{\mathbf{P}}_i\boldsymbol{\alpha}_i$;
  $\mathbf{r}_{i+1} = \mathbf{r}_i - \mathbf{Q}_i\boldsymbol{\alpha}_i$;
  $\mathbf{Z}_{i+1} = \mathbf{H}\mathbf{r}_{i+1}$;
  for $s = 1, \ldots, N$ **do**
    $t_i^s = \dfrac{\langle \mathbf{\Pi}\tilde{\mathbf{P}}_i\boldsymbol{\alpha}_i, \mathbf{Q}_i^s\boldsymbol{\alpha}_i\rangle}{\mathbf{r}_{i+1}^\top\mathbf{H}^s\mathbf{r}_{i+1}}$;
    if $t_i^s < \tau$ **then** `// local τ-test`
      $\mathbf{Z}_{i+1} = [\mathbf{Z}_{i+1} \,|\, \mathbf{H}^s\mathbf{r}_{i+1}]$;
      Subtract $\mathbf{H}^s\mathbf{r}_{i+1}$ from the first column in $\mathbf{Z}_{i+1}$;
    end
  end
  $\mathbf{\Phi}_{i,j} = {\mathbf{Q}_j}^\top\mathbf{Z}_{i+1}$; $\boldsymbol{\beta}_{i,j} = \mathbf{\Delta}_j^\dagger\mathbf{\Phi}_{i,j}$ for each $j = 0, \ldots, i$;
  $\tilde{\mathbf{P}}_{i+1} = \mathbf{Z}_{i+1} - \sum_{j=0}^{i} \tilde{\mathbf{P}}_j\boldsymbol{\beta}_{i,j}$;
end
Return $\mathbf{\Pi}\mathbf{x}_{i+1}$;

5 Numerical Results with FreeFem++ [18] and GNU Octave [10]

In this section we consider the linear system arising from applying BDD to a two dimensional linear elasticity problem discretized by standard $\mathbb{P}_1$ finite elements. The computational domain Ω is the unit square and we apply zero Dirichlet boundary conditions on the left hand side and zero Neumann elsewhere. The right hand side corresponds to a $(0, 10)^\top$ gravity vector and the initial guess is $\mathbf{0}$. For each test case we will specify the partition into N subdomains. This partition will be either regular (into squares of size $1/\sqrt{N} \times 1/\sqrt{N}$) or generated automatically by the graph partitioner Metis [22]. The mesh is a regular triangular mesh with $200 \times N$ elements. 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) [29, 38, 26, 36] or it is equal to the inverse of the multiplicity of the degree of freedom (multiplicity scaling). There are two physical parameters: Young's modulus E and Poisson's ratio ν. We set $\nu = 0.4$ in all of Ω for all test cases. For Young's modulus we choose a checkerboard distribution (see Figure 1 – left) where the two values are $E_1 = 10^7$ and, unless specified otherwise, $E_2 = 10^{12}$.

We compare five algorithms for solving the linear system: Algorithm 4 (New algorithm with Global τ–test), Algorithm 5 (New algorithm with Local τ–test), Algorithm 4 with $\tau = \infty$ (Simultaneous), Algorithm 1 where $\mathbf{U}$ is full rank and satisfies (12) (PPCG) and Algorithm 1 where $\mathbf{U}$ is full rank and satisfies (14) (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 methods and GenEO) we choose $\tau = 0.1$.

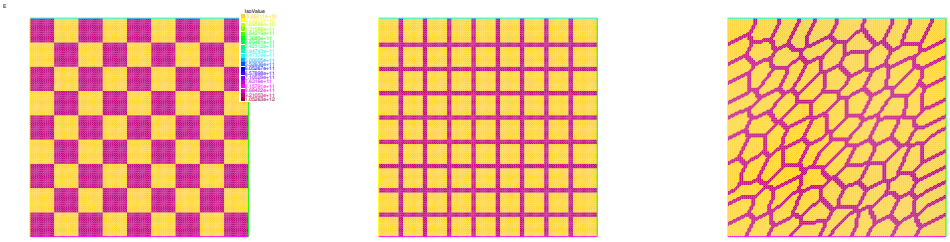

Figure 1: Left: Young's modulus; Center: Regular subdomains; Right: Metis subdomains.

5.1 Regular partition: Homogeneous Subdomains

The geometry for this test case is presented in Figure 1. The partition is into $N = 81$ subdomains (see Figure 1 – left) in such a way that the material parameters are constant in each subdomain. The results are presented in Figure 2. We examine three quantities with respect to the iteration count: the error $\|\mathbf{x}_i - \mathbf{x}_*\|_{\mathbf{A}}/\|\mathbf{x}_*\|_{\mathbf{A}}$, the dimension of the minimization space (see Property 2 in Theorem 1) and the number of local solves (number of applications of $\mathbf{S}^s$ or $\mathbf{S}^{s\dagger}$). In the case of GenEO we never include results on the number of local solves since this would require counting the number of iterations needed for the eigensolver to converge.

We observe that when k-scaling is used, convergence of PPCG is very good and the adaptive algorithms do not need to do any extra work (or very little in the case of the new algorithm with the local τ–test which selects just 4 extra search directions). This is what was expected since theoretical results [28, 26, 45] guarantee that PPCG converges fast. The simultaneous algorithm converges even faster but at the cost of about twice as many local solves.

When multiplicity scaling is used, convergence of PPCG is not as good. Both of the new algorithms adapt by selecting significantly larger minimization spaces (626 and 783 versus 268). This allows to reduce 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 we did not include it.

5.2 Metis partition: Heterogeneous Subdomains

Figure 3 shows the results when solving the same test case as previously with k-scaling and a Metis partition into $N = 81$ subdomains (see Figure 1 – right). The partition into subdomains is no longer aligned with the heterogeneities and domains are non regular. These are two known difficulties for BDD. On the left hand side we have plotted the results for all test cases and on the right hand side without PPCG so as to make the differences more clear between the competitive methods. The convergence of PPCG (top left plot in Figure 1) is a typical behaviour for an ill conditioned CG method [46]: first there is a long stagnation and then, once the isolated eigenvalues have been well approximated by the Ritz values, convergence is very fast. For all the other methods this stagnation has been completely eliminated (GenEO) or significantly decreased so this confirms that the problematic eigenvectors are being well handled by augmenting the minimization space with local quantities. The new methods select larger minimization spaces but only perform this augmentation during the first few iterations, then they behave as PPCG methods and the result is that they require roughly 4 times fewer local solves. On this hard test case the Simultaneous algorithm also convergences faster and with fewer local solves than PPCG.

In Table 1 we first give details on the number of iterations and local solves needed for conver-

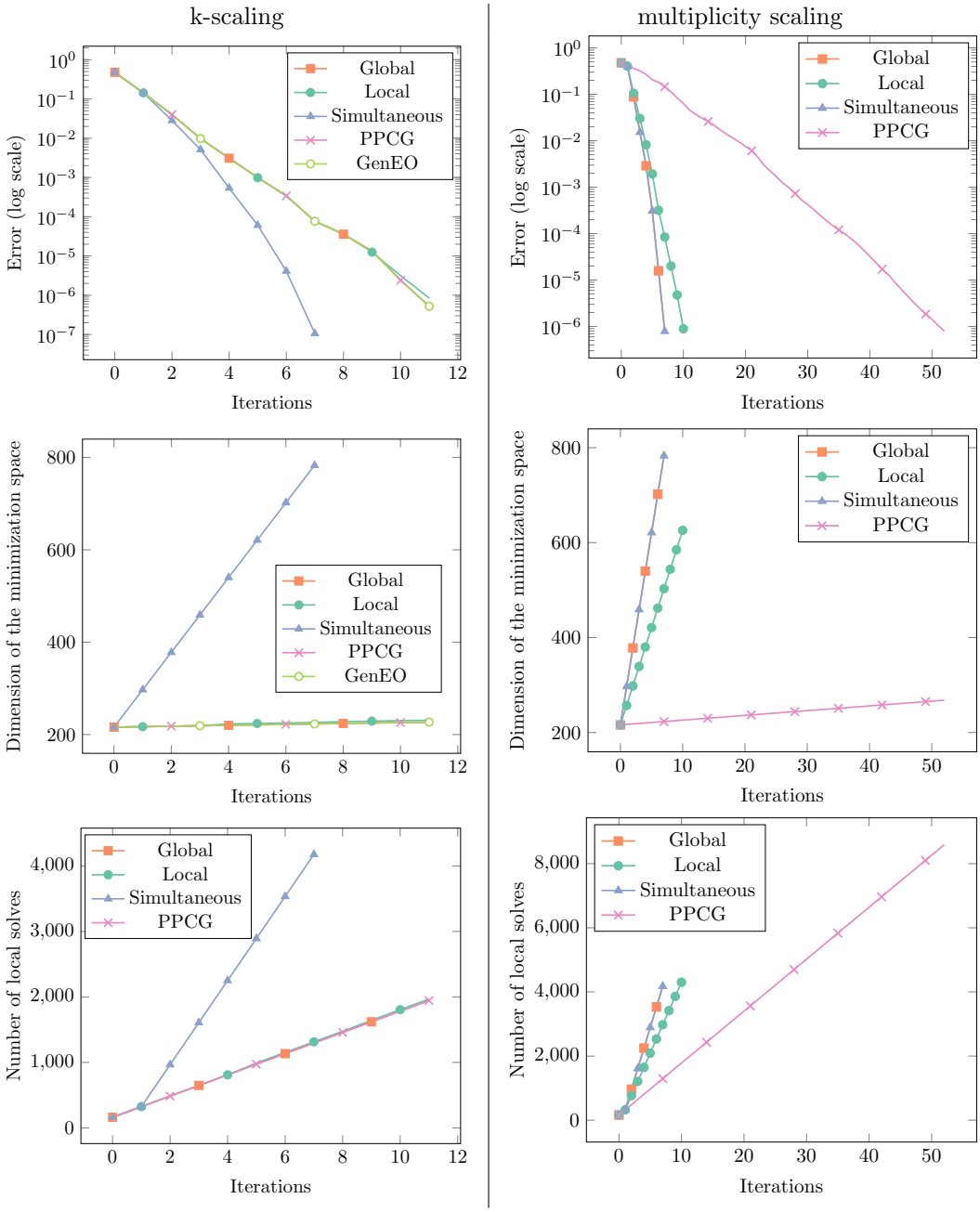

Figure 2: Regular partition. Left: k-scaling. Right: multiplicity scaling.

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

	Global τ–test		Local τ–test		Simultaneous		PCG		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		PCG		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		PCG		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.

gence when the heterogeneity E_2/E_1 varies (both for k-scaling and multiplicity scaling). Then we set $E_2/E_1 = 10^5$ and we make the number N of Metis subdomains vary. The number of squares in the checkerboard is also N. 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 not good in terms of parallelism.

For all methods the number of local solves also becomes larger as the problems become harder but the impact is a lot less important for the two new methods so we are satisfied with the way that they adapt to the difficulty. One final argument in favour of the new methods 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 [15] for the Simultaneous algorithm.

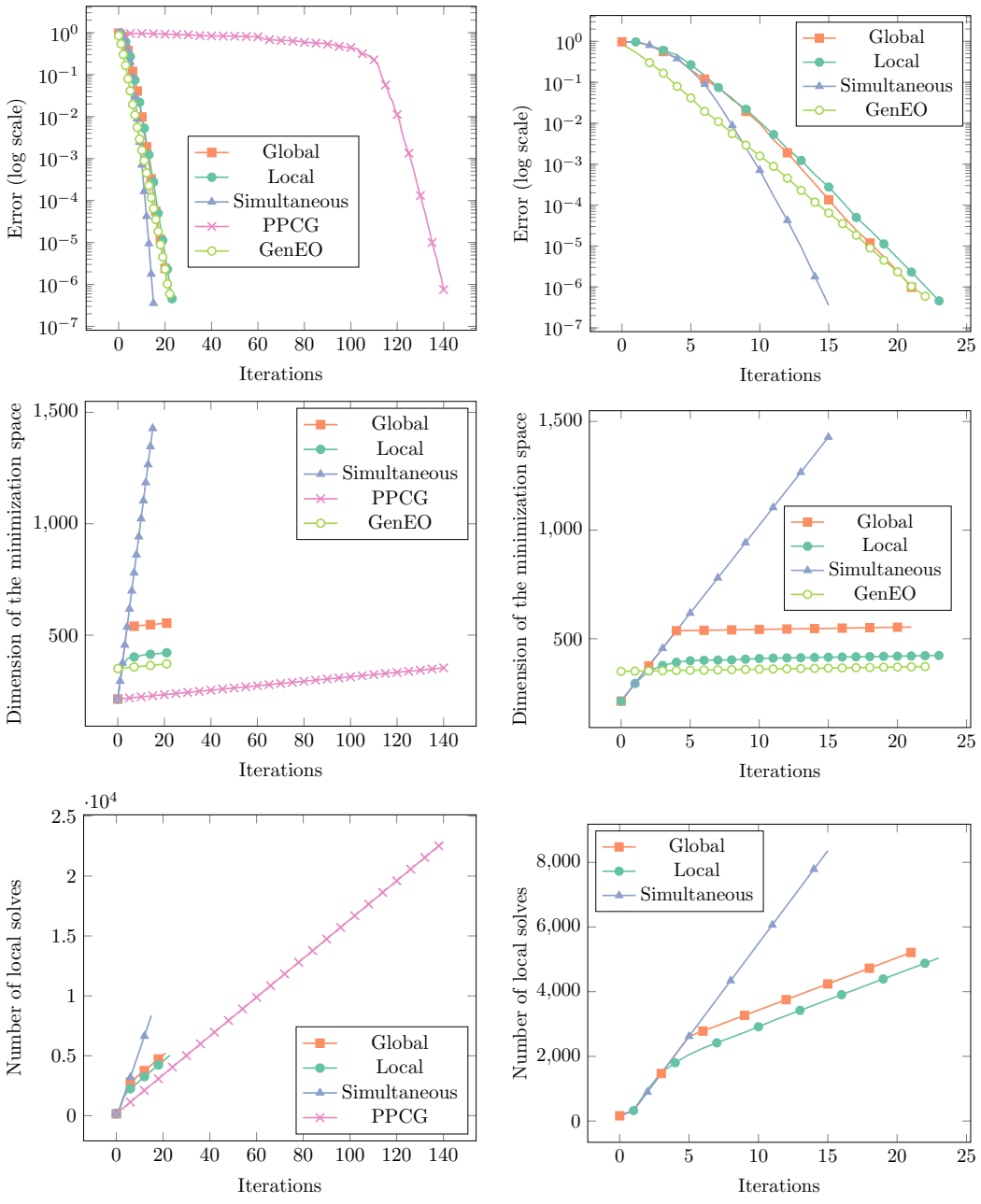

Figure 3: Metis Partition. Left: all methods. Right: zoom on competitive methods.

6 Conclusion and Perspectives

We have proposed a new adaptive solver (and a variant) for linear systems with multiple preconditioners. The theoretical analysis guarantees that there are two kinds of iterations: either the convergence is satisfying and an iteration of PPCG is performed or convergence is too slow and the minimization space is enriched with components coming from several preconditioners. We observed good convergence when applying the solver to the system arising from Balancing Domain Decomposition. This algorithm should now be implemented in parallel and CPU times should be measured to confirm efficiency. We believe that the new algorithms are good parallel solvers for industrial applications.

Future work includes applying the algorithm to other linear systems such as ones with several preconditioners each one arising 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.

Acknowledgements

The author would like to thank Pierre Gosselet, Frédéric Nataf, Daniel J. Rixen and François-Xavier Roux for some enlightening discussions on this topic.

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 preconditioners 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] 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.

[15] P. Gosselet, D. Rixen, F.-X. Roux, and N. Spillane. Simultaneous FETI and block FETI: Robust domain decomposition with multiple search directions. *International Journal for Numerical Methods in Engineering*, pages n/a–n/a, 2015.

[16] C. Greif, T. Rees, and D. Szyld. Additive Schwarz with variable weights. In J. Erhel, M. J. Gander, L. Halpern, G. Pichot, T. Sassi, and O. Widlund, editors, *Domain Decomposition Methods in Science and Engineering XXI*, volume 98 of *Lecture Notes in Computational Science and Engineering*, pages 779–787. Springer International Publishing, 2014.

[17] R. Haferssas, P. Jolivet, and F. Nataf. A robust coarse space for Optimized Schwarz methods SORAS-GenEO-2. Technical report, Submitted, hal-01100926, 2015.

[18] 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.

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

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

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

[22] 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.

[23] 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.

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

[25] 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.

[26] 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.

[27] S. Loisel, H. Nguyen, and R. Scheichl. Optimized schwarz and 2-lagrange multiplier methods for multiscale pdes. Technical report, submitted, 2015.

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

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

[30] 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.

[31] 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.

[32] 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.

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

[34] 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.

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

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

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

[38] 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.

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

[40] 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. Iterative methods for solving systems of algebraic equations (Copper Mountain, CO, 1998).

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

[42] N. Spillane. *Robust domain decomposition methods for symmetric positive definite problems.* PhD thesis, Thèse de l'Ecole doctorale de Mathmatiques de Paris centre, Laboratoire Jacques Louis Lions, Université Pierre et Marie Curie, Paris, 2014.

[43] 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.

[44] 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.

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

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