Reducing the communication and computational costs of Enlarged Krylov subspaces Conjugate Gradient

Laura Grigori, Olivier Tissot

NUMERICAL LINEAR ALGEBRA WITH APPLICATIONS
Numer. Linear Algebra Appl. 0000; **00**:1–24
Published online in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/nla

Reducing the communication and computational costs of Enlarged Krylov subspaces Conjugate Gradient

Laura Grigori[1]*Olivier Tissot[1]

[1] *Inria Paris, Alpines and UPMC Univ Paris 6, CNRS UMR 7598, Laboratoire Jacques-Louis Lions, Paris*

SUMMARY

In this paper we propose an algebraic method in order to reduce dynamically the number of search directions during block Conjugate Gradient iterations. Indeed, by monitoring the rank of the optimal step α_k it is possible to detect inexact breakdowns and remove the corresponding search directions. We also propose an algebraic criterion that ensures in theory the equivalence between our method with dynamic reduction of the search directions and the classical block Conjugate Gradient. Numerical experiments show that the method is both stable, the number of iterations with or without reduction is of the same order, and effective, the search space is significantly reduced.
We use this approach in the context of enlarged Krylov subspace methods which reduce communication when implemented on large scale machines. The reduction of the number of search directions further reduces the computation cost and the memory usage of those methods. Copyright © 0000 John Wiley & Sons, Ltd.

Received ...

KEY WORDS: linear solvers; block Krylov methods; Conjugate Gradient; inexact breakdowns

1. INTRODUCTION

Solving linear systems of the form $Ax = b$, with $A \in \mathbb{R}^{n \times n}$, is a problem that arises in many academic and industrial applications. We consider in this paper the case of symetric definite positive matrices, *i.e.* $z^\top Ay$ is a scalar product in $\mathbb{R}^n$, and Krylov subspace methods for solving these systems, more specifically the Conjugate Gradient method by *Hestenes and Stiefel* [5]. This method has been extensively studied over the years [20, 2].

Solving those linear systems efficiently on large scale computers remains a difficult problem. One challenge is the increased cost of communication with respect to computation [30], which shows that it is important to design or reformulate existing algorithms such that the number of communication instances is significantly reduced. One approach to do so with Krylov subspaces methods is to use Block Krylov subspaces which are larger spaces that can be constructed at the same communication cost as the original one, while leading to a faster convergence. Thus the overall number of communication instances can be drastically reduced.

Those were introduced for the Conjugate Gradient in 1980 by *O'Leary* with the Block Conjugate Gradient method [3] (Block CG). She was motivated by problems which require to solve a linear

*Correspondence to: Laura Grigori, Inria Paris, Alpines and UPMC Univ Paris 6, CNRS UMR 7598, Laboratoire Jacques-Louis Lions, Paris. E-mail: laura.grigori@inria.fr

Contract/grant sponsor: This work is funded by the NLAFET project as part of European Union's Horizon 2020 research and innovation program; contract/grant number: 671633

system with several right hand sides. She has shown theoritically that this method can converge significantly faster than the classical Conjugate Gradient.

In the early 90s *Nikishin and Yeremin* [7] introduced a novel method based on the Block Conjugate Gradient to solve a linear system with one right hand side (BRRHS-CG). They propose to reduce the size of the block during the iterations by monitoring the rank of the residual matrix. That way they aim at obtaining the convergence behaviour of the block method while maintaining an acceptable overcost compared to classical Conjugate Gradient. In the late 90s, *Brezinski* introduced *Multi-parameter Descent Methods* [13, 14]. Instead of using one search direction at each iteration he proposed to use several. This variant is a hybrid between a block method and a method for solving systems with one right hand side: even if the descent directions and the steps are blocks, as in the block methods, the remaining quantities are vectors, as in classical methods.

More recently *Robbé and Sadkane* in [4] improved the idea introduced by *Nikishin and Yeremin* [7] and applied it to GMRES [26]. With a different point of view *Bhaya et al.* [6] introduced the Cooperative Conjugate Gradient (Coop-CG) where several agents cooperate in order to solve a linear system. This method can be seen as Block CG that uses several random initial guesses [8].

Our starting point is the work of *Grigori, Moufawad and Nataf* from [8] where they introduce the *Enlarged Krylov* subspaces. They enrich the Krylov subspaces by splitting the residual obtained from the first iteration. In particular, we focus on the *Short Recurrence Enlarged CG* (EK-CG) method introduced in [8] because it keeps the short recurrence property of the classical Conjugate Gradient. And in fact, this method can be seen as a *Multi-parameter Descent Method* [13, 14]. The main difference lies in the approach used to construct the descent directions P_{k+1} at iteration $k+1$. In [14] the authors constructed P_{k+1} from the residual matrix R_k and the descent directions P_k from iteration k,

$$P_{k+1} = R_k - P_k(P_k^\top AP_k)^{-1}P_k^\top AR_k. \tag{1}$$

This is very close to the original formulation of Block Conjugate Gradient [3] and Conjugate Gradient [5]. In [8] the authors A-orthonormalize the blocks and construct P_{k+1} from the previous descent directions P_k and P_{k-1},

$$P_{k+1} = AP_k - P_kP_k^\top AAP_k - P_{k-1}P_{k-1}^\top AAP_k. \tag{2}$$

This is a modification of the original Block Lanczos formula,

$$P_{k+1} = AP_k - P_kP_k^\top AP_k - P_{k-1}P_{k-1}^\top AP_k, \tag{3}$$

in order to obtain P_k which are A-orthonormal to each others instead of being orthonormal.

In this paper we propose a general method to reduce the size of the search directions block during the iterations of a Block CG-like method. Unlike the method presented by *Nikishin and Yeremin* in [7], which assumes that P_k is constructed as in (1), we suppose that P_k is obtained using (2). We also propose an algebraic criterion to decide when to reduce the block size. Both the criterion and the reducing method are general and we apply them to BRRHS-CG, Coop-CG, and EK-CG. Numerical results show that this method is both effective (the size of the space in which we find the approximated solution is reduced by a factor of one third) and robust (the number of iterations increases by a factor smaller than 5%). Moreover we observe that EK-CG is particularly adapted to this method, it leads to the best results in terms of efectiveness and robustness in most of the numerical tests performed.

The paper is organised as follows. In section 2 we recall the definition and some basic properties of the Block CG method. We recall two variants in order to construct the descent directions, *Orthomin* which is the one used in the original CG method [5], and *Orthodir* which is based on the Lanczos formula (3).

In section 3 we formulate three methods to solve a linear system with one right hand side using the framework of Block CG: BRRHS-CG [7], Coop-CG [6] and EK-CG [8].

In section 4 we use the same idea as *Robbé and Sadkane* [4] and adapt their method to reduce dynamically the number of search directions to block CG, both for Orthodir and Orthomin variants. Indeed, by monitoring the rank of $\alpha_k = P_k^\top R_{k-1}$ we are able to reduce the size of the block during

DOI: 10.1002/nla

the iterations. This method is very general and can be applied to all Block CG-like methods. This allows us to compare BRRHS-CG, Coop-CG and EK-CG with block size reduction.

In section 5 we study the choice of the tolerance for removing the search directions. A theoritical study gives us an algebraic criterion that ensures the equivalence of the method with the one when no reduction is done. From this study, we derive a practical choice that induces no overcost and verifies the theoritical criterion. This choice does not depend on the method and therefore can be applied to all the methods presented before.

In section 6 we present numerical experiments. First, we compare the Orthodir and Orthomin variants. Numerical results show that Orthodir is always more stable in terms of number of iterations. Then we compare BRRHS-CG, Coop-CG and EK-CG with the classical Preconditioned Conjugate Gradient for the Orthodir variant. We observe that Ek-CG and BRRHS-CG behave similarly and outperform Coop-CG in terms of number of iterations. Finally we compare those 3 methods with block size reduction. Numerical results show that our method is stable when using Orthodir, the number of iterations is increasing slowly when we reduce the size of the block. But when using Orthomin, some untabilities may appear, the number of iterations increases drastically for some of our test matrices (from a linear elasticity problem).

2. THE BLOCK CG ALGORITHM AND THE ORTHODIR VARIANT

In this section we recall the definition of the Block CG algorithm according to *O'Leary* [3], also referred to as Orthomin [2]. Then we define the so-called Orthodir [2] variant of the method which leads to a faster convergence in terms of iteration count (see results in section 6).

In what follows we consider the following notations: $U^\top$ is the transpose of a matrix U, A is a symetric ($A^\top = A$) positive definite ($x^\top Ax > 0$, $\forall x \neq 0$) real matrix of size $n \times n$, B is a real matrix of size $n \times t$, $B^{(i)}$ is the i-th column of a matrix B, X_0 is an initial guess for the linear system $AX = B$, *i.e.* it is a real matrix of size $n \times t$. We denote the initial residual matrix $R_0 = B - AX_0$. We call t the initial block size. Unless otherwise stated, $||.||$ denotes the usual euclidean norm both for vectors and matrices.

Following *Gutknecht at al.* [10] Block Krylov subspaces are defined as,

$$\mathcal{K}_k^\square(A, R_0) := \text{span}^\square\left\{R_0, AR_0, \ldots, A^{k-1}R_0\right\} \tag{4}$$

$$:= \left\{\sum_{s=0}^{k-1} A^s R_0 \gamma_s \text{ such that } \forall s \in \{0, \ldots, k-1\}, \gamma_s \in \mathbb{R}^{t\times t}\right\}. \tag{5}$$

When there is no ambiguity we denote $\mathcal{K}_k^\square(A, R_0)$ by $\mathcal{K}_k^\square$. Using this definition Block Krylov suspaces projection methods are defined as,

$$X_k \in \mathcal{K}_k^\square + X_0, \tag{6}$$

$$R_k \perp \mathcal{L}_k^\square, \tag{7}$$

where $\mathcal{L}_k^\square$ is a subspace which has the same size as $\mathcal{K}_k^\square$. The first equation (6) is called the subspace condition and the second one (7) is called the Petrov-Galerkin condition.

The Block Conjugate Gradient method is defined as the Block Krylov subspaces projection method, where A is symmetric positive definite and $\mathcal{L}_k^\square = \mathcal{K}_k^\square$. As a result of this projection process,

$$\phi(X_k) = \min_{X \in \mathcal{K}_k^\square} \phi(X), \tag{8}$$

where

$$\phi(X) = \frac{1}{2} X^\top AX - B^\top X, \tag{9}$$

$$\nabla\phi(X) = AX - B. \tag{10}$$

DOI: 10.1002/nla

As in gradient methods, the new solution at iteration k is defined as $X_k = X_{k-1} + P_k\alpha_k$, where P_k represent the descent directions and α_k is the step. One important property of the Block Conjugate Gradient is the A-orthonormality (or conjugacy) of the descent directions, that is $P_i^\top AP_j = 0$ when $i \neq j$. The classical version of Block CG defined by ***O'Leary*** [3] is given in Algorithm 1.

Following [8], in our experiments we A-orthonormalize the block of P_k by using Pre-CholQR [11]. This algorithm can be found in the appendix of this paper (Algorithm 6). This method is very similar to the one originaly proposed by *Hestenes and Stiefel* [5] because it constructs the new descent directions P_{k+1} using R_k and P_k.

However in practice we notice that this variant is less effective when R_k becomes rank defficient. We will discuss this in more details in section 6 where we will present the numerical results in Table III. Following *Dubrulle* [12] it is possible to improve this algorithm by performing a QR decomposition of the residual matrix R_k before constructing P_{k+1} but we do not test this method in this paper.

The block version of the Orthodir method defined in [2] is given in Algorithm 2. Unlike the previous variant in Algorithm 2, P_{k+1} is constructed using P_k and P_{k-1}. This corresponds to the Block Lanczos algorithm but with the inner product induced by A.

Algorithm 1 Block CG: orthomin

Require: A, B, X_0, k_{max}, $\varepsilon_{\text{solver}}$
Ensure: $||B - AX|| < \varepsilon_{\text{solver}}$ or $k = k_{\text{max}}$
$R_0 = B - AX_0$
$P_1 =$ A-orthonormalize(R_0)
$k = 1$
while $||R_{k-1}|| > \varepsilon_{\text{solver}}||B||$ and $k < k_{\text{max}}$ **do**
  $\alpha_k = P_k^\top R_{k-1}$
  $X_k = X_{k-1} + P_k\alpha_k$
  $R_k = R_{k-1} - AP_k\alpha_k$
  $P_{k+1} = R_k - P_kP_k^\top AR_k$
  $P_{k+1} =$ A-orthonormalize(P_{k+1})
  $k = k + 1$
end while

Algorithm 2 Block CG: orthodir

Require: A, B, X_0, k_{max}, $\varepsilon_{\text{solver}}$
Ensure: $||B - AX|| < \varepsilon_{\text{solver}}$ or $k = k_{\text{max}}$
$R_0 = B - AX_0$
$P_0 = 0$
$P_1 =$ A-orthonormalize(R_0)
$k = 1$
while $||R_{k-1}|| > \varepsilon_{\text{solver}}||B||$ and $k < k_{\text{max}}$ **do**
  $\alpha_k = P_k^\top R_{k-1}$
  $X_k = X_{k-1} + P_k\alpha_k$
  $R_k = R_{k-1} - AP_k\alpha_k$
  $P_{k+1} = AP_k - P_kP_k^\top AAP_k - P_{k-1}P_{k-1}^\top AAP_k$
  $P_{k+1} =$ A-orthonormalize(P_{k+1})
  $k = k + 1$
end while

Both Orthomin and Orthodir produce P_i that are A-orthonormal and belong to the same space. Hence, they are mathematically equivalent in exact arithmetic. Orthodir is twice as

DOI: 10.1002/nla

expensive as Orthomin but it has the nice property to produce P_{k+1} which is full rank before A-orthonormalization.

Proposition 1
If $\forall i \geq 0$, P_i is A-orthonormal and

$$P_{k+1} = AP_k - P_k P_k^\top AAP_k - P_{k-1} P_{k-1}^\top AAP_k, \tag{11}$$

then P_{k+1} is full rank.

Proof
The first part of the proof is by induction on k.

- If $k = 0$, it is true by assumption.
- If $k \in \{1, \ldots, n\}$, let us assume that the proposition is true. We proove by contradiction that it is still true at iteration $k+1$. We supppose that the proposition is true at iteration k and that there exists a vector v of size t such that,

$$P_{k+1} v = 0. \tag{12}$$

Then,

$$\begin{aligned} & (AP_k - P_k P_k^\top AAP_k - P_{k-1} P_{k-1}^\top AAP_k)\, v &= 0, & (13)\\ \iff \quad & (I - P_k P_k^\top A - P_{k-1} P_{k-1}^\top A) AP_k\, v &= 0, & (14)\\ \iff \quad & AP_k\, v &= 0. & (15) \end{aligned}$$

Hence, AP_k is not full rank but both A and P_k are full rank and there is a contradiction.

□

Proposition 1 ensures that using A-CholQR (Algorithm 5) with Orthodir is always possible. There is no breakdown in this case because P_{k+1} is already full rank before its A-orthonormalization. This is not true with Orthomin because the rank P_{k+1} depends of the one of R_k and R_k becomes rank deficient at some point. Numerical results shown in Table III illustrate this. In order to overcome this difficulty, it is possible to use Pre-CholQR (Algorithm 6) instead of A-CholQR, as shown in [11]. Even when using Pre-CholQR, Orthomin performs more iterations than Orthodir on very ill-conditioned matrices (Table III). In the following section we will define a process to reduce the size of the search directions during the iterations both with Orthomin and Orthodir.

3. USING BLOCK CG TO SOLVE ONE LINEAR SYSTEM OF EQUATIONS

Even if the Block CG method was initially used to solve linear systems with several right hand sides [3], it is possible to use a block method to solve a linear system with only one rigth hand side. This is useful because Block CG can converge significantly faster than CG [3]. In the following we present different ways to use Block CG in order to solve a system with only one right hand side.

Let us recall that A is $n \times n$ and X_0, B are $n \times t$. The block size is denoted by t. We denote $\mathbb{1}_t$ a row of size $1 \times t$ and full of ones. We refer to the method defined in [7] by *Nikishin and Yeremin* as ***BRRHS-CG***. In this method, X_0 and B are choosen as $X_0 = x_0 \mathbb{1}_t$ $B^{(i)} \sim \mathcal{U}(0,1)$, $\forall t \geq i > 1$ and $B^{(1)} = b$. In that case, the method is stopped as soon as the first column of R_k has a norm smaller than $\varepsilon_{\text{solver}}||b||$. The solution is given by the first column of X. ***Coop-CG*** is a method defined by *Bhaya et al.* in [6] in which X_0 is a uniform random matrix and $B = b\mathbb{1}_t$. As soon as one column of the residual R has a norm smaller than $\varepsilon_{\text{solver}}||b||$, the method is stopped. The solution is given by the corresponding column of X.

In [8] *Grigori, Moufawad and Nataf* use a domain decomposition approach to define enlarged Krylov subspaces and the associated ***EK-CG*** method. More precisely given a splitting

DOI: 10.1002/nla

decomposition represented by the operator T,

$$T(x) = \begin{pmatrix} * & & & & & & \\ \vdots & & & & & & \\ * & & & & & & \\ & * & & & & & \\ & \vdots & & & & & \\ & * & & & & & \\ & & \ddots & & & & \\ & & & * & & & \\ & & & \vdots & & & \\ & & & * & & & \\ & & & & * & & \\ & & & & \vdots & & \\ & & & & * & & \end{pmatrix}$$

and the initial residual r_0, the corresponding enlarged Krylov subspace is defined as

$$\mathcal{K}_{k,t} = \text{span}^{\square}\{T(r_0), AT(r_0), \ldots, A^{k-1}T(r_0)\}. \tag{16}$$

Using this definition and the formula (2) to construct the search directions, they derive a *Short Recurrence Enlarged* CG. This method can be embedded in the Block Conjugate Gradient framework by defining $R_0 = T(r_0)$. The stopping criterion is the euclidean norm of $r_k = \sum_i R_k^{(i)}$ and the solution is $x = \sum_i X^{(i)}$.

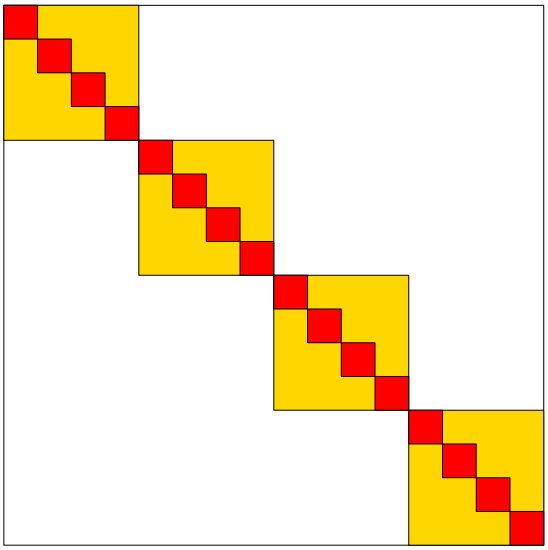
In our experiments we use a splitted block Jacobi preconditioner,

$$LL^\top = \begin{pmatrix} A_{11} & & & \\ & A_{22} & & \\ & & \ddots & \\ & & & A_{\text{nj nj}} \end{pmatrix}$$

with $t <$ nj, and we solve $L^{-1}AL^{-\top}x = L^{-1}b = \widetilde{b}$. In that case the matrix $\widetilde{A} = L^{-1}AL^{-\top}$ is still symmetric positive definite and we can apply all the presented CG variants we presented on the preconditioned system $\widetilde{A}x = \widetilde{b}$. As it is shown in the figure (1), we consider a number of blocks nj which is different (and higher in general) than the block size of the method.

DOI: 10.1002/nla

Figure 1. Pattern of the matrix: in red the block Jacobi and in yellow the subdomains corresponding to the partitioning of r_0.



It is also possible to apply the preconditioner on the left, we solve $M^{-1}Ax = M^{-1}b$ with $M = LL^\top$. Although numerical results show that the two approaches are equivalent (Table VII), there are some pratical differences. When preconditioning on the left, another vector is needed and the A-orthonormalization is done with respect to A without preconditioning (see Algorithms 7 and 8). This second point is of importance because when using splitted preconditioner A-orthonormalization is done with respect to the preconditioned matrix which is more stable numerically [11].

4. REDUCING THE BLOCK SIZE

In this section, we introduce an approach for reducing the block size in the Orthodir method during the iterations. This is a technique known as deflation in block Krylov methods [10, 7, 4].

Indeed, as explained in the survey [10] the key idea to reduce the block size is to monitor the rank of R_{k-1}. Once R_{k-1} becomes rank deficient, it means that there exists a vector v of dimensions $t \times 1$ such that,

$$R_{k-1}v = 0, \tag{17}$$

and,

$$\begin{aligned} R_k v &= R_{k-1}v - AP_k\alpha_k v && (18)\\ &= 0 + AP_kP_k^\top R_{k-1}v && (19)\\ &= 0. && (20) \end{aligned}$$

It follows that $R_i v = 0$ for $i \geq k-1$. In other words, $X_{k-1}v$ has already converged at iteration $k-1$ because $X_{k-1}v = A^{-1}Bv$. For $i \geq k-1$, there exists a linear combination (independant of i) of columns of X_i denoted $X_i v$ such that $X_i v$ remains constant. As a consequence, it is possible to follow [7] and reduce effectively the sizes of X, R and P. But as Langou showed in [15], it can lead to unstabilities. It is easy to see, using exactly the same reasoning, that if $\text{rank}(R_{k-2}) - \text{rank}(R_{k-1}) = l$ then a part of the solution of dimension l has converged. As R_{k-1} is an $n \times t$ matrix with n large, it is preferable to avoid computing the rank of R_{k-1} directly. Our approach is based on computing the rank of $\alpha_k = P_k^\top R_{k-1}$. This is similar to the idea developped by *Robbé and Sadkane* in [4]: it is preferable to work with the residual projected onto the Krylov

DOI: 10.1002/nla

subspace because it is smaller ($t \times t$ instead of $n \times t$) and rank(α_k) = rank(R_{k-1}). Indeed, if α_k is rank deficient there exists v such that,

$$\alpha_k v = 0 \implies P_k^\top R_{k-1} v = 0, \tag{21}$$

$$\implies P_k^\top u = 0 \text{ with } u = R_{k-1} v, \tag{22}$$

and since P_k is always full rank after its A-orthonormalization,

$$P_k^\top u = 0 \implies u = 0, \tag{23}$$

$$\implies R_{k-1} v = 0. \tag{24}$$

Thus rank(α_k) = rank(R_{k-1}) and $R_i v = 0$ for $i \geq k-1$. It follows that $\alpha_i v = 0$ for $i \geq k-1$ which means that some search directions are not taken into account anymore. However, in pratice this case, where α_k becomes exactly rank deficient (also denoted *exact breakdown* or *lucky breakdown*), is very rare and it is preferable to detect when α_k becomes nearly rank deficient (also denoted *inexact breakdown*) [10, 4, 7].

More precisely, we compute the Singular Value Decomposition of α_k,

$$\alpha_k = U_k \Sigma_k V_k^\top, \tag{25}$$

where U_k and V_k are orthonormal $t \times t$ matrices and Σ =diag$(\sigma_t, \ldots, \sigma_1)$ ($\sigma_1 \leq \cdots \leq \sigma_t$ are the singular values of α_k). If α_k is nearly rank deficient then this decomposition can be rewritten as

$$U_k \Sigma_k V_k^\top = \begin{pmatrix} U_k^{(1)} & U_k^{(2)} \end{pmatrix} \begin{pmatrix} \Sigma_k^{(1)} & 0 \\ 0 & \Sigma_k^{(2)} \end{pmatrix} \begin{pmatrix} {V_k^{(1)}}^\top \\ {V_k^{(2)}}^\top \end{pmatrix}, \tag{26}$$

where $||U_k^{(2)} \Sigma_k^{(2)} {V_k^{(2)}}^\top|| < \varepsilon_{\text{def}}$, and ε_{def} is a given tolerance. In this case $\alpha_k \approx U_k^{(1)} \Sigma_k^{(1)} {V_k^{(1)}}^\top$ and the idea is to replace α_k by $U_k^{(1)} \Sigma_k^{(1)} {V_k^{(1)}}^\top$. Hence $P_k \alpha_k \approx (P_k U_k^{(1)})(\Sigma_k^{(1)} {V_k^{(1)}}^\top)$.

Now let us define,

$$P_k^{(1)} = P_k U_k^{(1)}, \tag{27}$$

$$P_k^{(2)} = P_k U_k^{(2)}, \tag{28}$$

$$\alpha_k^{(1)} = \Sigma_k^{(1)} {V_k^{(1)}}^\top. \tag{29}$$

Theorem 1
The part of the solution corresponding to $X_k V_k^{(2)}$ has almost converged in the A-norm. More precisely, if we denote the error $E_k = X^* - X_k$ (of size $n \times t$) where X^* is the exact solution, and v a column of $V_k^{(2)}$,

$$||E_k\, v - E_{k-1}\, v||_A \leq \varepsilon_{\text{def}}, \tag{30}$$

$$||R_k\, v - R_{k-1}\, v||_{A^{-1}} \leq \varepsilon_{\text{def}}. \tag{31}$$

Proof
Let us assume that v is the j^{th} column of $V_k^{(2)}$. We denote σ_j the corresponding singular value, it corresponds to the j^{th} diagonal value in $\Sigma_k^{(2)}$. We have,

$$||X_k\, v - X_{k-1}\, v||_A = ||X_{k-1}\, v + P_k U_k^{(2)} \sigma_j\, v - X_{k-1}\, v||_A \tag{32}$$

$$= ||P_k^{(2)} \Sigma_k^{(2)}\, v||_A \tag{33}$$

$$= \sqrt{\sigma_j v^\top {P_k^{(2)}}^\top A P_k^{(2)} v\, \sigma_j} \tag{34}$$

$$= \sqrt{\sigma_j^2} \tag{35}$$

$$= \sigma_j \tag{36}$$

$$\leq \varepsilon_{\text{def}}. \tag{37}$$

DOI: 10.1002/nla

Similarly,

$$
\begin{aligned}
||R_k\, v - R_{k-1}\, v||_{A^{-1}} &= ||R_{k-1}\, v + AP_k U_k^{(2)} \sigma_j\, v - R_{k-1}\, v||_{A^{-1}} && (38)\\
&= ||AP_k^{(2)} \Sigma_k^{(2)} v||_{A^{-1}} && (39)\\
&= \sqrt{\sigma_j v^\top {P_k^{(2)}}^\top A A^{-1} A P_k^{(2)} v\, \sigma_j} && (40)\\
&= \sqrt{\sigma_j^2} && (41)\\
&= \sigma_j && (42)\\
&\leq \varepsilon_{\text{def}}. && (43)
\end{aligned}
$$

□

Since the part of the solution corresponding to $X_k V_k^{(2)}$ has almost converged (in the A-norm), the search directions $P_k^{(2)}$ are not needed anymore. Hence we define the new search directions as,

$$
\begin{aligned}
P_{k+1} U_k^{(1)} &= AP_k^{(1)} - \begin{pmatrix} P_k^{(1)} & P_k^{(2)} \end{pmatrix} \begin{pmatrix} {P_k^{(1)}}^\top \\ {P_k^{(2)}}^\top \end{pmatrix} AAP_k^{(1)} && (44)\\
&\quad -P_{k-1} P_{k-1}^\top AAP_k^{(1)} \\
&= AP_k^{(1)} - P_k^{(1)} {P_k^{(1)}}^\top AAP_k^{(1)} && (45)\\
&\quad -P_k^{(2)} {P_k^{(2)}}^\top AAP_k^{(1)} - P_{k-1} P_{k-1}^\top AAP_k^{(1)}
\end{aligned}
$$

and $P_{k+1} U_k^{(1)}$ have a smaller size than P_{k+1}. Furthermore, by construction $P_{k+1} U_k^{(1)}$ belongs to span$^\square\{AP_k^{(1)}\}$ and is A-orthogonal to $P_0, P_1, \ldots, P_{k-1}, P_k^{(1)}, P_k^{(2)}$.

This allows us to define the new search directions the first time we reduce the size of the block but we need to generalize this idea when the size is reduced several times (possibly until it is equal to one).

To do so we denote H the matrix formed by the removed search directions $P_i^{(2)}$, $i \in \{1, \ldots, n\}$. It is important to note that if α_k is full rank, then $P_k^{(2)}$ is empty. Thus the number of columns of H is at most $t-1$ where t is the initial block size. Indeed, the final block size is at least 1 which means that the columns of H are the $t-1$ search directions removed. $P_i^{(1)}$ is formed by the search directions that are used at iteration i if α_i is nearly rank deficient. In order to prove Proposition 2 we will use the following lemma.

Lemma 1
For all $k \in \{1, \ldots, n\}$ we have:

$$
{P_k^{(1)}}^\top AP_k^{(2)} = 0 \tag{46}
$$

Proof

$$
\begin{aligned}
{P_k^{(1)}}^\top AP_k^{(2)} &= {U_k^{(1)}}^\top P_k^\top AP_k U_k^{(2)} && (47)\\
&= {U_k^{(1)}}^\top U_k^{(2)} && (48)\\
&= 0 && (49)
\end{aligned}
$$

□

We want to construct P_{k+1} such that:

$$
{P_{k+1}}^\top AP_i^{(d)} = 0,\, i < k+1,\, d = \{1, 2\}, \tag{50}
$$

DOI: 10.1002/nla

and $P_{k+1} \in \text{span}^{\square}\{AP_k^{(1)}\}$.

One way to achieve this is to use Proposition 2. It leads to a variant of Orthodir where the number of search directions can be reduced dynamically during the iterations (Algorithm 3).

Another possibility is to use Proposition 3. Unlike Proposition 2 the new search directions are constructed using Orthomin instead of Orthodir (Algorithm 4). But it also allows to reduce the number of search directions dynamically during the iterations.

Proposition 2
If we define P_{k+1} such that,

$$\begin{aligned} P_0 &= 0, && (51)\\ P_1 &= \text{A-orthonormalize}(R_0), && (52)\\ P_{k+1} &= AP_k^{(1)} - P_k^{(1)}{P_k^{(1)}}^{\top} AAP_k^{(1)} && (53)\\ &\quad -P_{k-1}^{(1)}{P_{k-1}^{(1)}}^{\top} AAP_k^{(1)} - HH^{\top} AAP_k^{(1)}. \end{aligned}$$

Then it is such that,

$$ {P_{k+1}}^{\top} AP_i^{(d)} = 0,\ i \leq k,\ d = \{1, 2\}. \quad (54)$$

Proof
The proof is based on induction on k.

- If $k \in \{0, 1\}$ the descent directions P_k are the same as in the usual Block Conjugate Gradient and (54) is true.
- If $k \in \{2, \ldots, n\}$ we assume that (54) is true for k and $P_i^{(1)}$, $i \leq k$. We want to prove that it remains true for P_{k+1}. For all $i \leq k$, $d = \{1, 2\}$ we have:

$$\begin{aligned} {P_i^{(d)}}^{\top} AP_{k+1} &= {P_i^{(d)}}^{\top} AAP_k^{(1)} && (55)\\ &- {P_i^{(d)}}^{\top} AP_k^{(1)}{P_k^{(1)}}^{\top} AAP_k^{(1)} && (56)\\ &- {P_i^{(d)}}^{\top} AP_{k-1}^{(1)}{P_{k-1}^{(1)}}^{\top} AAP_k^{(1)} && (57)\\ &- {P_i^{(d)}}^{\top} AHH^{\top} AAP_k^{(1)} && (58) \end{aligned}$$

  - If $d = 1$ then (58) is vanishing because ${P_i^{(1)}}^{\top} AP_l^{(2)} = 0$: whether because $i \neq l$, or they are equal and we can use Lemma (1) to conclude that ${P_i^{(1)}}^{\top} AP_i^{(2)} = 0$. If $i = k$ then (57) is equal to zero and (56) can be rewritten as:

$$ {P_k^{(d)}}^{\top} AP_k^{(1)}{P_k^{(1)}}^{\top} AAP_k^{(1)} = {P_k^{(1)}}^{\top} AAP_k^{(1)}. \quad (59)$$

    So (55) = (56) and it follows that (54) is true. The same reasoning holds if $i = k - 1$. If $k \leq k - 2$ then (56) and (57) vanish. It remains to show that (55) is equal to zero. This is true because by construction $P_k^{(1)}$ is A-orthogonal to $\text{span}^{\square}\left\{P_0, P_1, \ldots, P_{k-1}^{(1)}, H\right\}$ so ${P_i^{(d)}}^{\top} AAP_k^{(1)} = 0$. Hence (54) is true.
  - If $d = 2$ then (56) and (57) are vanishing because whether $i \leq k - 2$, or $i \in \{k - 1, k\}$ and we can use Lemma (1) to conclude. But if $d = 2$ this means that we reduced the size at iteration i and consequently (58) can be rewritten as:

$$\begin{aligned} {P_i^{(2)}}^{\top} AHH^{\top} AAP_k^{(1)} &= {P_i^{(2)}}^{\top} AP_i^{(2)}{P_i^{(2)}}^{\top} AAP_k^{(1)} && (60)\\ &= {P_i^{(2)}}^{\top} AAP_k^{(1)} && (61) \end{aligned}$$

    So (55) = (58) and it follows that (54) is true.

DOI: 10.1002/nla

Algorithm 3 Orthodir with block size reduction

H

Require: A, B, X_0, $k_{\max}$, $\varepsilon_{\text{solver}}$, ε_{def}
Ensure: $||B - AX|| < \varepsilon_{\text{solver}}$ or $k = k_{\max}$
$R_0 = B - AX_0$
$P_0 = 0$
$P_1 = $ A-orthonormalize(R_0)
$k = 1$
$H = 0$
while $||R_{k-1}|| < \varepsilon_{\text{solver}}||B||$ and $k < k_{\max}$ **do**
  $\alpha_k = P_k^\top R_{k-1}$
  $[U_k, \Sigma_k, V_k] = \text{svd}(\alpha_k)$
  $s_k =$ number of singular values of α_k bigger than ε_{def}
  if $s_k < s_{k-1}$ **then**
    $U_k^{(1)} = U_k(:, 1 : s_k)$
    $U_k^{(2)} = U_k(:, s_k + 1 : end)$
    $\Sigma_k^{(1)} = \Sigma_k(1 : s_k, 1 : s_k)$
    $V_k^{(1)} = V_k(:, 1 : s_k)$
    $P_k^{(2)} = P_k U_k^{(2)}$
    $H = [H, P_k^{(2)}]$
    $\alpha_k = \Sigma_k^{(1)} {V_k^{(1)}}^\top$ ▷ Reduce α_k size
    $P_k = P_k U_k^{(1)}$ ▷ Reduce P_k size
  end if
  $X_k = X_{k-1} + P_k \alpha_k$
  $R_k = R_{k-1} - AP_k\alpha_k$
  $P_{k+1} = AP_k - P_k P_k^\top AAP_k - P_{k-1}P_{k-1}^\top AAP_k - HH^\top AAP_k$
  $P_{k+1} = $ A-orthonormalize(P_{k+1})
  $k = k + 1$
end while

□

Proposition 3
Let P_{k+1} be defined such that,

$$P_0 = 0, \tag{62}$$
$$P_1 = \text{A-orthonormalize}(R_0), \tag{63}$$
$$P_{k+1} = R_k - P_k^{(1)} {P_k^{(1)}}^\top AR_k. \tag{64}$$

Then it is such that,

$${P_{k+1}}^\top AP_i^{(d)} = 0,\ i \leq k,\ d = \{1, 2\}. \tag{65}$$

Proof
The proof is by induction on k.

- If $k \in \{0, 1\}$ the descent directions P_k are the same as in the usual Block Conjugate Gradient and (65) is true.

DOI: 10.1002/nla

- If $k \in \{2, \dots, n\}$ we assume that (65) is true for k and $P_i^{(1)}$, $i \leq k$. We want to prove that it remains true for P_{k+1}. For all $i \leq k$, $d = \{1, 2\}$ we have:

$$\begin{aligned} {P_i^{(d)}}^\top A P_{k+1} &= {P_i^{(d)}}^\top A R_k \quad (66) \\ &- {P_i^{(d)}}^\top A P_k^{(1)} {P_k^{(1)}}^\top A R_k \end{aligned}$$

  - If $(i, d) \neq (k, 1)$ then,

$$\begin{aligned} {P_i^{(d)}}^\top A P_{k+1} &= {P_i^{(d)}}^\top A R_k \quad (67) \\ &= 0, \quad (68) \end{aligned}$$

    by construction of R_k.
  - If $(i, d) = (k, 1)$ then,

$$\begin{aligned} {P_i^{(d)}}^\top A P_{k+1} &= {P_k^{(1)}}^\top A R_k - {P_k^{(1)}}^\top A R_k \quad (69) \\ &= 0. \quad (70) \end{aligned}$$

□

Algorithm 4 Orthomin with block size reduction

H

Require: A, B, X_0, k_{max}, $\varepsilon_{\text{solver}}$, ε_{def}
Ensure: $||B - AX|| < \varepsilon_{\text{solver}}$ or $k = k_{\text{max}}$

```
R_0 = B − AX_0
P_0 = 0
P_1 = A-orthonormalize(R_0)
k = 1
while ||R_{k−1}|| < ε_solver||B|| and k < k_max do
    α_k = P_k^⊤ R_{k−1}
    [U_k, Σ_k, V_k] = svd(α_k)
    s_k = number of singular values of α_k bigger than ε_def
    if s_k < s_{k−1} then
        U_k^(1) = U_k(:, 1 : s_k)
        Σ_k^(1) = Σ_k(1 : s_k, 1 : s_k)
        V_k^(1) = V_k(:, 1 : s_k)
        α_k = Σ_k^(1) V_k^(1)^⊤                ▷ Reduce α_k size
        P_k = P_k U_k^(1)                      ▷ Reduce P_k size
    end if
    X_k = X_{k−1} + P_k α_k
    R_k = R_{k−1} − AP_k α_k
    P_{k+1} = R_k − P_k^⊤ P_k A R_k
    P_{k+1} = A-orthonormalize(P_{k+1})
    k = k + 1
end while
```

If $\alpha_{k+1} = {P_{k+1}}^\top R_k$ is nearly rank deficient then we replace it by its low rank approximation and $P_{k+1}^{(1)} = P_{k+1} U_{k+1}^{(1)}$ and $P_{k+1}^{(2)} = P_{k+1} U_{k+1}^{(2)}$. Otherwise the size is not reduced and $P_{k+1}^{(2)}$ is empty.

5. DEFLATION TOLERANCE

In this section we study the choice of the deflation tolerance in Algorithms 3 and 4. Let us recall that in [25] the authors show that $||Z||_2 \leq |||A^{-1}||_2^{1/2}$ if Z is A-orthonormal. It is interesting to note

DOI: 10.1002/nla

that,

$$||\alpha_k||_2 = ||P_{k+1}^\top R_k||_2 \tag{71}$$
$$\leq ||P_{k+1}||_2 ||R_k||_2 \tag{72}$$
$$\leq ||A^{-1}||_2^{1/2} ||R_k||_2. \tag{73}$$

Hence, the following proposition holds.

Proposition 4
If we choose ε_{def} such that

$$\varepsilon_{\text{def}} < ||A^{-1}||_2^{1/2} \varepsilon_{\text{solver}} ||B||_2, \tag{74}$$

then in exact arithmetic, Algorithms 3 and 4 are equivalent to their static counterpart Algorithms 2 and 1.

Proof
If α_k does not loose full rank then it is clear that the proposition is true. On the other hand, if α_k is rank deficient, using Theorem 1, there exits $v \in \mathbb{R}^n$ such that for all $k \geq k^*$,

$$||E_k\, v - E_{k-1}\, v||_A \leq \varepsilon_{\text{def}}. \tag{75}$$

And,

$$||E_k\, v - E_{k-1}\, v||_A \geq ||A^{-1}||_2^{1/2} ||E_k\, v - E_{k-1}\, v||_2. \tag{76}$$

Putting the two inequations together,

$$||E_k\, v - E_{k-1}\, v||_2 \leq ||A^{-1}||_2^{-1/2} \varepsilon_{\text{def}} \tag{77}$$
$$\leq \varepsilon_{\text{solver}} ||B||_2. \tag{78}$$

This means that $X_k\, v$ has already converged at the required tolerance. Hence, this part of the solution will remain constant during the following iterations. □

Corollary 1
In Algorithms 3 and 4, if 74 is verified, the number of search directions used at iteration k, denoted s_k, is decreasing, *i.e.* $1 \leq s_{k+1} \leq s_k \leq t\ \forall k \in \{1, \ldots, n\}$.

Proof
As stated in the proof of Proposition 4, $X_k\, v$ has already converged when removing the search directions related to v for all the following iterations. As the size s_k corresponds to t minus the number of such vectors v. It remains to show that s_k is always larger than 1. Let us assume that it is zero. Then we would have $||\alpha_k||_2 = \varepsilon_{\text{def}}$ and if 74 is verified then the method would have converged. □

Remark 1
In Algorithms 3 and 4 the converged solution is at most $\varepsilon_{\text{solver}}$ accurate even if we perform more iterations after the stopping criterion. After reducing the size we cannot expect higher accuracy than $\varepsilon_{\text{solver}}$ because we removed some information based on $\varepsilon_{\text{solver}}$ which is fixed *a priori* in the method. In other words, if ε_{def} is too large the method will not converge (and if it is too small the number of search directions will not decrease).

Proof
This is a direct application of Theorem 1. Once we removed search directions we cannot recover them because we construct the new one by A-orthonormalizing against all previous ones. □

Remark 2
The space $\mathcal{K}_k^{\triangle}$ is not exactly the same as $\mathcal{K}_k^{\square}$. In fact, it is smaller because some of the search directions have been removed. Even if Proposition 4 is true in exact arithmetic, due to round-off errors during orthonormalization we are expecting to do (hopefully not many) more iterations when reducing the size.

DOI: 10.1002/nla

Proposition 5
When using the block Conjugate Gradient to solve a linear system with a single right hand side, the following deflation criterion ensures that the methods with dynamic reduction of the search directions are equivalent in exact arithmetic to their static counterpart,

$$\varepsilon_{\text{def}} < \frac{1}{\sqrt{t}}||A^{-1}||_2^{1/2}\varepsilon_{\text{solver}}||b||_2. \tag{79}$$

Proof
When solving a system with a single right hand side,

$$\begin{aligned} ||r_k||_2 &= ||R_k\mathbb{1}_t||_2 && (80)\\ &\leq ||R_k||_2||\mathbb{1}_t||_2 && (81)\\ &\leq ||R_k||_2\sqrt{t}. && (82)\end{aligned}$$

Hence,

$$\begin{aligned} ||\alpha_k||_2 &\leq ||A^{-1}||_2^{1/2}||R_k||_2 && (83)\\ &\leq \frac{1}{\sqrt{t}}||A^{-1}||_2^{1/2}||r_k||_2 && (84)\\ &\leq \frac{1}{\sqrt{t}}||A^{-1}||_2^{1/2}\varepsilon_{\text{solver}}||b||_2. && (85)\end{aligned}$$

The rest of the proof is exactly the same as in the one of Proposition 4. □

Remark 3
In practice, it is too costly to compute the optimal criteria in (74) and (79). But it is easy to see that if,

$$||A^{-1}||_2^{1/2} \geq 1, \tag{86}$$

then we can use,

$$\varepsilon_{\text{def}} = \varepsilon_{\text{solver}}||b||_2, \tag{87}$$

instead of 74, and,

$$\varepsilon_{\text{def}} = \frac{1}{\sqrt{t}}\varepsilon_{\text{solver}}||b||_2, \tag{88}$$

instead of 79.

In all the numerical experiments, the criterion for removing the number of search directions is 88.

6. NUMERICAL RESULTS

All the results are obtained with Matlab R2015b. PCG is the Matlab Preconditioned Conjugate Gradient method. We always use a splitted block Jacobi preconditioner and we refer to nj as the number of diagonal blocks used.

We compare the performance of the three methods on a set of matrices that are also used in [8, 21, 22] where they are described in more details. These matrices are displayed in Table I where we present their size, the number of nonzeros, their smallest and largest eigenvalues. The matrices NH2D, SKY2D, SKY3D and ANI3D arise from boundary value problem of the diffusion equations:

$$\begin{aligned} -\text{div}(\kappa(x)\nabla u) &= f && \text{on } \Omega && (89)\\ u &= 0 && \text{on } \partial\Omega_D && (90)\\ \frac{\partial u}{\partial n} &= 0 && \text{on } \partial\Omega_N && (91)\end{aligned}$$

DOI: 10.1002/nla

where Ω is the unit square (2D) or cube (3D). The tensor κ is a given coefficient of the partial differential operator. In the 2D case $\partial\Omega_D = [0,1] \times \{0,1\}$ and in the 3D case $\partial\Omega_D = [0,1] \times \{0,1\} \times [0,1]$. In both cases, $\partial\Omega_N$ is choosen as $\partial\Omega_N = \partial\Omega \setminus \partial\Omega_D$.

The matrix NH2D is obtained by considering a nonhomogeneous problem with large jumps in the coefficients of κ. The tensor κ is isotropic and discontinuous, it jumps from the constant value 10^3 in the ring $\frac{1}{2\sqrt{2}} \leq |x-c| \leq \frac{1}{2}$ with $c = (\frac{1}{2}, \frac{1}{2})^\top$, to 0 outside.

The matrices SKY2D and SKY3D are obtained by considering skyscraper problems where the domain contains many zones of high permeability wich are isolated from each other. More precisely κ is taken as:

$$\kappa(x) = 10^3 * ([10 * x_2] + 1) \qquad \text{if } [10x_i] \text{ is odd, } i = \{1,2\} \tag{92}$$

$$\kappa(x) = 1 \qquad \text{otherwise,} \tag{93}$$

where $[x]$ is the integer value of x.

The matrix ANI3D is obtained by considering anisotropic layers: the domain is made of 10 anisotropic layers with jumps of up to four orders of magnitude and an anisotropy ratio of 10^3 in each layer. Those layers are parallel to $z = 0$, of size 0.1, and inside them the coefficients are constant: $\kappa_y = 10\kappa_x$, $\kappa_z = 100\kappa_x$.

All those problemes are discretized on cartesian grids, of size 100×100 for the 2D problems and of size $20 \times 20 \times 20$ for the 3D problems.

The Ela matrices arise from the linear elasticity problem with Dirichlet and Neumann boundary conditions defined as follows

$$\text{div}(\sigma(u)) + f = 0 \qquad \text{on } \Omega \tag{94}$$

$$u = 0 \qquad \text{on } \partial\Omega_D \tag{95}$$

$$\sigma(u) \cdot n = 0 \qquad \text{on } \partial\Omega_N \tag{96}$$

Ω is a unit square (2D) or cube (3D). The matrices Ela$N$ correspond to this equation discretized using a triangular mesh with $N \times 10 \times 10$ points on the corresponding vertices. The matrices Ela2DN correspond to this equation discretized unsing a triangular mesh with $N \times N$ points on the corresponding vertices. $\partial\Omega_D$ is the Dirichlet boundary, $\partial\Omega_N$ is the Neumann boundary, f is some body force, u is the unknown displacement field. $\sigma(.)$ is the Cauchy stress tensor given by Hooke's law: it can be expressed in terms of Young's Modulus E and Poisson's ration ν. For a more detailed description of the problem see [23] and [9]. We consider discontinuous E and ν in 3D: $(E_1, \nu_1) = (2 \times 10^{11}, 0.25)$ and $(E_2, \nu_2) = (10^7, 0.45)$; and discontinuous E in 2D were the fluid is nearly incompressible: $(E_1, \nu_1) = (10^{12}, 0.45)$ and $(E_2, \nu_2) = (2 \times 10^6, 0.45)$. Those matrices are scaled in order to reduce the effect of possibly very high values on the diagonal.

Table I. Test matrices we use in our tests, their size, the number of nonzeros, their smallest (λ_{min}) and largest (λ_{min}) eigenvalues; if they are coming from 2D or 3D discretization and the type of problem they are coming from.

	Size	Nonzeros	λ_{min}	λ_{max}	2D/3D	Problem
NH2D	10 000	49 600	1.9e-3	8.0	2D	Boundary Value
SKY2D	10 000	49 600	3.5e-3	7.0e4	2D	Skyscraper
SKY3D	8 000	53 600	5.3e-3	3.0e3	3D	Skyscraper
ANI3D	8 000	53 600	6.7e-7	1.4	3D	Anisotropic Layers
Ela25	9 438	312 372	2.7e-5	3.4	3D	Linear Elasticity P1 FE
Ela50	18 153	618 747	1.9e-6	3.4	3D	Linear Elasticity P1 FE
Ela100	36 663	1 231 497	2.6e-7	2.4	3D	Linear Elasticity P1 FE
Ela2D200	80 802	964 800	2.8e-8	3.7	2D	Linear Elasticity P1 FE

DOI: 10.1002/nla

Table II. Numerical parameters we consider in our tests, the size of the initial block (which is also the number of right hand sides), the number of diagonal blocks nj in the block diagonal preconditioner, the variant we use between Orthomin and Orthodir, the definition of the error and the number of iterations..

t	block size
nj	number of block Jacobi
Odir	Orthodir
Omin	Orthomin
er	$\frac{\|\|x^*-x_k\|\|_2}{\|\|x^*\|\|_2}$ with x^* the exact solution (computed with Matlab LU)
iter	number of iterations until convergence

In Table III we summarize the results obtained when running Orthomin (Algorithm 1) and Orthodir (Algorithm 2) on the same matrix for BRRHS-CG [7], Coop-CG [6], and EK-CG [8]. We denote Omin1 the version where we use A-CholQR (Algorithm 5) to A-orthonormalize the search directions block and Omin2 the version where we use Pre-CholQR (Algorithm 6). We also compare with Matlab PCG method. We observe that breakdowns happen with Omin1 when the matrix is very ill-conditioned (Ela100). Omin2 may perform 30% more iterations than Orthodir on difficult matrices (Ela50 and Ela100) but they are equivalent on simple test cases (NH2D, SKY2D, SKY3D, ANI3D) and there is no breakdowns unlike Omin1. In all the following experiments we always use Omin2 and denote it as Omin.

Table III. Number of iterations to get the solution ($\varepsilon_{\text{solver}} = 10^{-8}$), the stopping criterion is the error, *i.e.* $\frac{\|\|x^*-x_k\|\|_2}{\|\|x^*\|\|_2}$ where x^* is the exact solution computed with LU. Omin2 and Odir are equivalent on NH2D, SKY2D, SK3D and ANI3D but on elasticity test cases (Ela50 and Ela100) Odir is more stable than Omin2. Omin1 is the less stable because breakdowns can occur on difficult matrices.

			PCG	BRRHS-CG			Coop-CG			EK-CG		
	t	nj	Omin	Odir	Omin1	Omin2	Odir	Omin1	Omin2	Odir	Omin1	Omin2
NH2D	16	64	98	38	38	38	41	41	41	34	34	34
SKY2D	16	64	388	46	68	68	50	70	70	43	43	43
SKY3D	16	64	311	52	52	52	53	54	54	51	51	51
ANI3D	16	64	77	62	62	62	65	65	65	58	58	58
Ela50	16	64	483	88	88	88	106	120	120	94	94	94
Ela100	16	64	485	67	-	114	85	-	127	73	-	118

The results presented in Table IV compare the classical PCG, BRRHS-CG, and EK-CG on several matrices from our test set. In the first three columns we present experiments where we increase both the number of block Jacobi nj and the block size t while keeping their ratio constant. As the number of block Jacobi increases, the preconditioner becomes less effective. Our goal is to compensate this by an increase of the block size so that the number of iterations does not increase or even decreases. The corresponding results are summerized in the first columns of Table IV. In all the cases considered, BRRHS-CG and EK-CG behave very similarly: the number of iterations is almost the same for both methods. When the block size increases, and even if the preconditioner is less effective, the block methods can drastically reduce the number of iterations with respect to PCG (except for ANI3D for which we observe that the iteration count obtained with 2 vectors in the block is reduced by a factor 2 and up to 3 when 32 vectors are used in the block). Furthermore, the block methods are very effective in comparison with PCG. This is especially true when the block size is large (a factor more than 2 and up to more than 10). But even for small block sizes we can observe a significant gain, nearly a factor of 2 for the elasticity matrices for example.

The last three column in Table IV present results obtained when the block size is increased while the number of blocks nj in the block Jacobi preconditioner is kept constant. The corresponding results are summerized in the last 4 columns of Table IV. In this case we expect that the number

DOI: 10.1002/nla

of iterations will decrease because the preconditioner is the same but the block size increases, so the search space grows faster with larger block size. And this is what we obtain. For almost all matrices (except ANI3D) we observe a gain of a factor between 3 and 10 in terms of iteration count. However, for several matrices up to a certain block size the gain is not significant anymore (NH2D and SKY2D for example). As in the previous experiment, BRRHS-CG and EK-CG behave very similarly in terms of iteration count.

Table IV. Number of iterations to get the solution ($\varepsilon_{\text{solver}} = 10^{-8}$) with Orthodir (Algortihm 2, and PCG is the usual Preconditioned Conjugate Gradient), the stopping criterion is the error, *i.e.* er $= \frac{||x^* - x_k||_2}{||x^*||_2}$ where x^* is the exact solution computed with LU.

	t	nj	PCG	BRRHS-CG	EK-CG	nj	PCG	BRRHS-CG	EK-CG
NH2D	2	8	67	51	51	256	128	102	103
	4	16	77	47	48	256	128	78	78
	8	32	86	43	43	256	128	63	59
	16	64	98	38	34	256	128	48	45
	32	128	113	33	30	256	128	38	34
	64	256	128	28	26	256	128	28	26
SKY2D	2	8	165	95	98	256	493	259	257
	4	16	209	74	70	256	493	129	130
	8	32	328	59	55	256	493	97	74
	16	64	388	46	43	256	493	48	46
	32	128	428	38	36	256	493	37	35
	64	256	493	33	32	256	493	33	32
SKY3D	2	8	165	104	103	256	417	348	337
	4	16	217	79	79	256	417	272	250
	8	32	322	90	90	256	417	155	156
	16	64	314	50	50	256	417	84	83
	32	128	386	40	41	256	417	47	48
	64	256	417	30	31	256	417	30	31
ANI3D	2	8	55	53	52	256	100	93	94
	4	16	66	59	58	256	100	92	88
	8	32	73	62	60	256	100	88	85
	16	64	78	64	61	256	100	83	78
	32	128	90	63	62	256	100	72	69
	64	256	100	58	56	256	100	58	56
Ela25	2	8	191	119	121	256	440	281	279
	4	16	220	95	107	256	440	188	206
	8	32	263	84	90	256	440	137	148
	16	64	309	72	76	256	440	102	104
	32	128	357	62	66	256	440	78	80
	64	256	440	58	62	256	440	58	62
Ela50	2	8	294	171	187	256	649	363	384
	4	16	364	134	145	256	649	235	248
	8	32	411	106	116	256	649	166	168
	16	64	485	88	94	256	649	115	125
	32	128	570	74	79	256	649	84	88
	64	256	649	63	69	256	649	63	69

The results in Table V show a comparison between BRRHS-CG, Coop-CG and EK-CG with (Algorithm 2) and without reducing the block size (Algorithm 3). We observe that our method to reduce the block size is stable in practice: the number of iterations with or without the block size reduction is of the same order, and still far lower than with the usual PCG method. More precisely the gain is between 22% less iterations for ANI3D test case, and up to a factor of 17 for the quasi incompressible elasticity test case. For most of the test cases, the block methods perform 5 to 15 less iterations than the usual PCG even when the block size is reduced. However, it is effective because

DOI: 10.1002/nla

H
Table V. Number of iterations to get the solution with Orthodir with dynamic reduction of the search directions (Algorithm 3, $\varepsilon_{\text{solver}} = 10^{-6}$ and $\varepsilon_{\text{def}} = \varepsilon_{\text{solver}}||b||_2 - \varepsilon_{\text{machine}}$ for BRRHS-CG and Coop-CG, and $\varepsilon_{\text{def}} = \frac{\varepsilon_{\text{solver}}||b||_2}{\sqrt{t}}$ for EK-CG), the stopping criterion is the normalized residual. The initial the block size is 32 and the number of block Jacobi is 1024. The symbol ✓ means that we reduced the size of the block and $\times$ means that we used the usual algorithm.

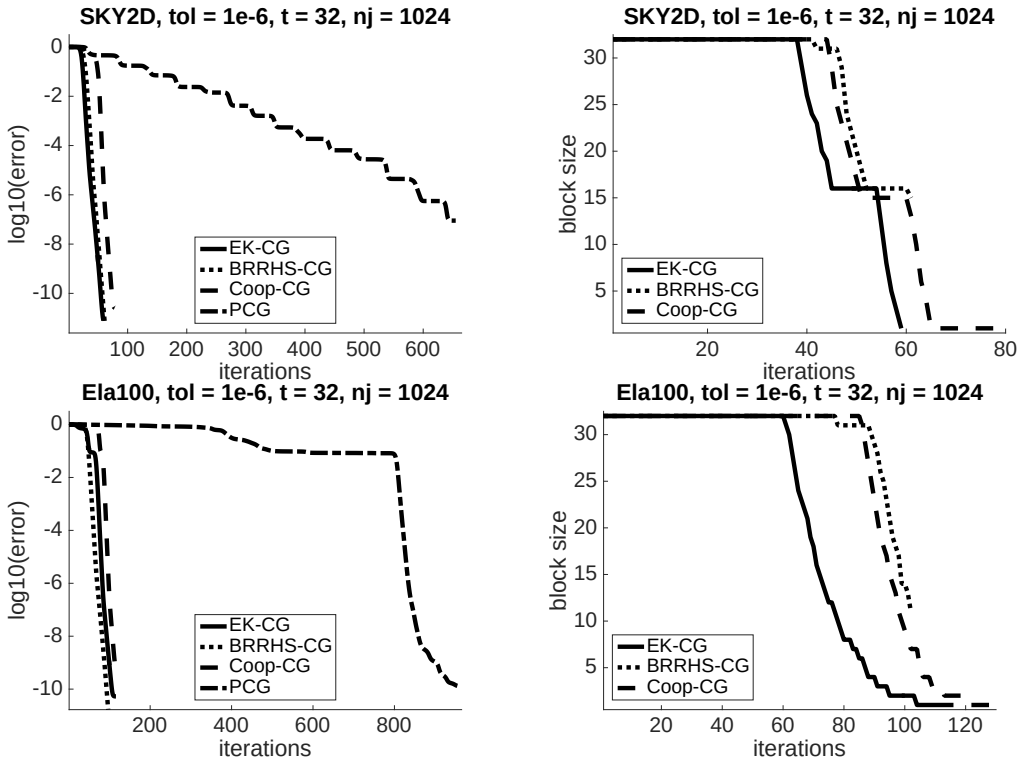
		PCG		BRRHS-CG			Coop-CG			EK-CG		
	red. size	iter	er	iter	er	dim($\mathcal{K}_k^{\triangle}$)	iter	er	dim($\mathcal{K}_k^{\triangle}$)	iter	er	dim($\mathcal{K}_k^{\triangle}$)
NH2D	$\times$	179	5.3e-08	52	3.2e-09	1664	58	2.3e-09	1856	47	4.0e-09	1504
	✓	179	5.3e-08	52	3.4e-09	1609	58	3.3e-09	1631	48	3.0e-09	1391
SKY2D	$\times$	655	9.0e-08	61	6.4e-12	1952	75	3.5e-11	2400	57	6.9e-12	1824
	✓	655	9.0e-08	61	7.6e-12	1739	78	2.6e-11	1739	59	7.8e-12	1546
SKY3D	$\times$	428	3.1e-07	71	2.0e-09	2272	77	1.7e-09	2464	69	1.3e-09	2208
	✓	428	3.1e-07	71	4.3e-09	2153	81	9.0e-10	2189	69	4.8e-09	2032
ANI3D	$\times$	111	1.7e-07	85	1.3e-07	2720	87	1.3e-07	2784	84	1.4e-07	2688
	✓	111	1.7e-07	85	1.3e-07	2711	88	1.1e-07	2645	85	1.8e-07	2321
Ela100	$\times$	955	1.3e-10	102	5.1e-12	3264	125	4.8e-12	4000	109	3.2e-11	3488
	✓	955	1.3e-10	102	5.2e-12	3093	128	5.8e-10	3084	116	5.3e-11	2384
Ela2D200	$\times$	4551	1.2e-09	255	1.4e-10	8160	301	2.2e-10	9632	253	1.8e-10	8096
	✓	4551	1.2e-09	258	3.3e-10	7331	327	5.6e-10	6986	266	1.4e-10	6553

the dimension of the search space (denoted $\mathcal{K}_k^{\triangle}$) is well decreased. For the test cases where the number of iteration is relatively small (as in NH2D, SKY2D, SKY3D and ANI3D test cases), the dimension of the search space is descreased by around 10%. But for the more complex test cases as elasticity test cases, the dimension of the search space is reduced between 20% and 30%. For those test cases, EK-CG performs very well. The final search space is smaller but the number of iterations is still very low. For the Ela100 test case, the search space of EK-CG is around 25% smaller than the final one of BRRHS-CG and Coop-CG, and the number of iterations is about 10% higher than BRRHS-CG, but around 10% lower than Coop-CG.

In Figure 2, we plot the error (left), as well as the size of the block (right), as a function of the number of iterations for SKY2D and Ela100. We consider that the initial block size is 32 and the number of blocks nj in the block Jacobi preconditioner is 1024. For both matrices, we observe that the convergence of the error is far better for the block methods compared to PCG even with the block size reduction (a factor of 7 for SKY2D and 10 for Ela100). Still there is a small plateau before convergence for all the methods which is a well known phenomenon with Krylov methods [10, 18]. The block size reduction and the error behave similarly for both matrices, the block size is reduced when the system is starting to converge (around iteration 40 for SKY2D and iteration 60 for Ela100). For the two test cases, EK-CG performs better than Coop-CG and BRRHS-CG, it reduces its search directions faster than Coop-CG and better than BRRHS-CG. For these tests, unlike Coop-CG and EK-CG, BRRHS-CG final block size is greater than 1. But it keeps a convergence speed similar as the one of BRRHS-CG, and always faster than the one of Coop-CG.

DOI: 10.1002/nla

Figure 2. Block size reduction and error decreasing as a function of the number of iterations when using Orthodir with dynamic reduction of the search directions (Algorithm 3). The right figures represent the plot of the error as a function of the number of iterations. We use a log_{10} scale for the error. The left figures represent the plot of the block size as a function of the number of iterations. We only plot the block size for the block methods and not for PCG.



In short, the results in Table V and Figure 2 show that EK-CG finds the solution of the linear system in a smaller subspace than the other block methods considered (BRRHS-CG and Coop-CG) when reducing the search directions with Algorithm 3.

The results in Table VI show a comparison between BRRHS-CG, Coop-CG and EK-CG with (Algorithm 1) and without reducing the block size (Algorithm 4). In that case we observe that our method to reduce the block size is not always stable in practice. Although the number of iterations for NH2D, SKY2D, SKY3D and ANI3D is almost the same when reducing or not the number of search directions, it is not the case for elasticity matrices where the reduction can lead to very slow convergence. However, for the matrices NH2D, SKY2D, SKY3D and ANI3D we observe the similar behaviour as with Algorithm 3. The blocks methods converge faster with or without reducing the number of search directions are faster than PCG (between 5 and 6 times less iterations for SKY3D for example). For all those matrices (NH2D, SKY2D, SKY3D and ANI3D) EK-CG achieves both a fast convergence in terms of iterations and a good reduction of the search space. For example, for ANI3D when reducing the size EK-CG and BRRHS-CG have the same iteration count but the search space of EK-CG is around 10% smaller. For elasticity matrices, even if the method seems to converge (which is prooved in theory for the deflation criterion we choosed), the convergence is very slow in terms of iterations. Extra numerical experiments suggest that this is due to the loss of othogonality. Indeed, we tried to A-orthonormalize P_{k+1} against $P_k^{(2)}$ explicitely and this decreases the number of iterations. However, it is not possible to do it in practice because the extra cost in terms of flops and communications would prevent any improvement in terms of performance compared to Orthomin without dynamical reduction of the search directions.

DOI: 10.1002/nla

H
Table VI. Number of iterations to get the solution with Orthomin with dynamic reduction of the search directions (Algorithm 4, $\varepsilon_{\text{solver}} = 10^{-6}$ and $\varepsilon_{\text{def}} = \frac{\varepsilon_{\text{solver}}||b||_2}{\sqrt{t}}$), the stopping criterion is the nomalized residual. The initial the block size is 32 and the number of block Jacobi is 1024. The symbol ✓ means that we reduced the size of the block and × means that we used the usual algorithm.

		PCG		BRRHS-CG			Coop-CG			EK-CG		
	red. size	iter	er	iter	er	dim($\mathcal{K}_k^{\triangle}$)	iter	er	dim($\mathcal{K}_k^{\triangle}$)	iter	er	dim($\mathcal{K}_k^{\triangle}$)
NH2D	×	179	5.3e-08	52	3.2e-09	1664	58	2.3e-09	1856	47	4.0e-09	1504
	✓	179	5.3e-08	52	3.3e-09	1609	58	3.5e-09	1632	48	3.0e-09	1392
SKY2D	×	655	9.0e-08	61	4.2e-08	1952	263	2.7e-09	8416	57	8.1e-10	1824
	✓	655	9.0e-08	62	4.2e-08	1754	322	4.4e-09	2099	61	8.2e-10	1566
SKY3D	×	428	3.5e-07	71	3.5e-09	2272	80	1.8e-09	2560	69	2.4e-09	2208
	✓	428	3.1e-07	72	6.2e-09	2180	108	8.9e-08	2263	71	4.4e-09	2043
ANI3D	×	111	1.7e-07	85	1.3e-07	2720	87	1.3e-07	2784	84	1.4e-07	2688
	✓	111	1.7e-07	85	1.3e-07	2711	88	1.1e-07	2647	85	1.6e-07	2322
Ela100	×	955	1.3e-10	146	5.5e-10	4672	100	5.1e-07	3200	157	4.2e-10	5024
	✓	955	1.3e-10	322	3.4e-09	4485	100	1.1e-04	3178	+1000	-	-
Ela2D200	×	4551	1.2e-09	244	1.7e-06	7808	235	1.4e-04	7520	241	3.0e-08	7712
	✓	4551	1.2e-09	+1000	-	-	238	4.3e-04	7562	+1000	-	-

7. CONCLUSION

Starting from block Conjugate Gradient [3], we expressed a familly of methods for solving symmetric positive definite systems. A first difference is the method for constructing the search directions. Orthomin is the original method presented in [3], it follows the original Conjugate Gradient of Hestenes and Stiefel [5]. Following [8] it is also possible to construct the search directions using Orthodir, this method is presented in [2] in the case of Conjugate Gradient.

Although Orthodir is twice as expensive as Orthomin, it has the advantages to be breakdown-free without needing to use Pre-CholQR (Algorithm 6). Even if in theory they are equivalent if there is no breakdown, numerical results show that Orthodir is also more effective than Orthomin in terms of iterations on very ill-conditioned matrices.

We presented a method that allows to reduce dynamically the number of search directions during a block CG-like method. It is completely algebraic, and does not depend of the variant of block CG used (BRRHS-CG, Coop-CG, EK-CG), nor the formula for constructing the search directions (Orthomin or Orthodir). We also proposed a practical tolerance that ensures (in exact arthmetic) the equivalence of our method with the case where the number of search directions is not reduced. Numerical experiments show that the method used with Orthodir (Algorithm 3) is both stable, the number of iterations with or without the reduction of the number of search directions is of the same order, and effective, the search space is reduced from 10% and up to 30% in practice. However, the method used with Orthomin (Algorithm 4), is not stable on very ill-conditioned matrices. The number of iterations can dractically increase. On the other matrices, the behaviour of the method is almost the same as with Orthodir, and we observe that EK-CG is preferable compared to the other variants we studied because it allows to reduce the number of search directions effectively while maintaining a good convergence speed.

To conclude, we estimate that Orthomin is, in general, more effective than Orthodir if the matrix is not very ill-conditioned because it is equivalent to Orthodir but twice as cheap both in terms of flops and communication. For very ill-conditioned matrices, Orthodir with reduction of the number of search directions is more stable. As future work, the methods will be implemented in parallel and their performance will be evaluated on massively parallel machines.

DOI: 10.1002/nla

APPENDIX A

The two following algorithms are described in [11]. The first one, A-CholQR (Algorithm 5) is a generalization of CholQR to the case of an oblique inner product. The second one, Pre-CholQR (Algorithm 6) is a more robust version of A-CholQR that adds an extra QR factorization at the beginning.

Algorithm 5 AChol-QR(P)

Require: P
Ensure: $P^\top AP = \mathbb{I}_{t\times t}$
1: $C = P^\top AP$
2: $L = chol(C)$
3: $P = L/R$
4: **for** $i = 1 : t$ **do**
5: $\quad P^{(i)} = \frac{P^{(i)}}{||P^{(i)}||_A}$
6: **end for**

Algorithm 6 PreChol-QR(P)

Require: P
Ensure: $P^\top AP = \mathbb{I}_{t\times t}$
1: $P = QR$
2: $C = Q^\top AQ$
3: $L = chol(C)$
4: $P = L/R$
5: **for** $i = 1 : t$ **do**
6: $\quad P^{(i)} = \frac{P^{(i)}}{||P^{(i)}||_A}$
7: **end for**

APPENDIX B

Following Saad [27], the idea for applying left preconditioning to Conjugate Gradient is to remark that $M^{-1}A$ is self-adjoint with respect to the M-inner product. It is possible to generalize this idea to the block case by remarking that,

$$X^\top AM^{-1}MY = X^\top MM^{-1}AY. \tag{97}$$

Hence, if we denote by $Z_k = M^{-1}R_k$ the residual for the preconditioned system, it is possible to rewrite the Algoritms 1 and 2 (ignoring the initial step) for the matrix $M^{-1}A$ using the M-inner product instead of the euclidean one,

$$\alpha_k = P_k^\top MZ_{k-1} = P_k^\top R_{k-1}, \tag{98}$$
$$X_k = X_{k-1} + P_k\alpha_k, \tag{99}$$
$$R_k = R_{k-1} - AP_k\alpha_k, \tag{100}$$
$$Z_k = M^{-1}Z_k. \tag{101}$$

Then it is possible to derive the Orthomin version as,

$$P_{k+1} = Z_k - P_kP_k^\top AM^{-1}MZ_k \tag{102}$$
$$= Z_k - P_kP_k^\top AZ_k, \tag{103}$$

DOI: 10.1002/nla

or the Orthodir version as,

$$P_{k+1} = M^{-1}AP_k - P_kP_k^\top AM^{-1}MM^{-1}AP_k - P_{k-1}P_{k-1}^\top AM^{-1}MM^{-1}AP_k \quad (104)$$
$$= M^{-1}AP_k - P_kP_k^\top AM^{-1}AP_k - P_{k-1}P_{k-1}^\top AM^{-1}AP_k. \quad (105)$$

To derive those relationships we assumed that, similarly to the unpreconditioned case, the block P_k is $M^{-1}A$-orthonormal with respect to the M-inner product. More precisely,

$$P_k^\top MM^{-1}AP_k = P_k^\top AP_k = I. \quad (106)$$

Therefore, it is equivalent to A-orthonormalize the block P_k. Putting all thoses relationships together, it is possible to derive Algorithms 7 and 8.

Algorithm 7 Preconditioned Orthomin BCG

Require: A, M, B, X_0, k_{max}, $\varepsilon_{\text{solver}}$
Ensure: $||B - AX|| < \varepsilon_{\text{solver}}$ or $k = k_{\text{max}}$
$R_0 = B - AX_0$
$P_1 =$ A-orthonormalize$(M^{-1}R_0)$
$k = 1$
while $||R_{k-1}|| > \varepsilon_{\text{solver}}||B||$ and $k < k_{\text{max}}$ **do**
  $\alpha_k = P_k^\top R_{k-1}$
  $X_k = X_{k-1} + P_k\alpha_k$
  $R_k = R_{k-1} - AP_k\alpha_k$
  $Z_k = M^{-1}R_k$
  $P_{k+1} = Z_k - P_kP_k^\top AZ_k$
  $P_{k+1} =$ A-orthonormalize(P_{k+1})
  $k = k + 1$
end while

Algorithm 8 Preconditioned Orthodir BCG

Require: A, M, B, X_0, k_{max}, $\varepsilon_{\text{solver}}$
Ensure: $||B - AX|| < \varepsilon_{\text{solver}}$ or $k = k_{\text{max}}$
$R_0 = B - AX_0$
$P_0 = 0$
$P_1 =$ A-orthonormalize$(M^{-1}R_0)$
$k = 1$
while $||R_{k-1}|| > \varepsilon_{\text{solver}}||B||$ and $k < k_{\text{max}}$ **do**
  $\alpha_k = P_k^\top R_{k-1}$
  $X_k = X_{k-1} + P_k\alpha_k$
  $R_k = R_{k-1} - AP_k\alpha_k$
  $Z_k = M^{-1}AP_k$
  $P_{k+1} = Z_k - P_kP_k^\top AZ_k - P_{k-1}P_{k-1}^\top AZ_k$
  $P_{k+1} =$ A-orthonormalize(P_{k+1})
  $k = k + 1$
end while

DOI: 10.1002/nla

Table VII. Number of iterations to get the solution ($\varepsilon_{\text{solver}} = 10^{-6}$), the stopping criterion is the error, *i.e.* er $= \frac{||x^* - x_k||_2}{||x^*||_2}$ where x^* is the exact solution computed with LU . The comparison is between left or splitted preconditioner, and the results are the same for both.

			Odir		Omin	
	t	nj	Left	Split	Left	Split
NH2D	8	256	48	48	48	48
SKY2D	8	256	65	65	65	65
SKY3D	8	256	147	147	147	147
ANI3D	8	256	65	65	65	65
Ela50	8	256	149	149	149	149

REFERENCES

1. G. H. GOLUB AND C. F. VAN LOAN, *Matrix Computations*, Third ed., The Johns Hopkins University Press, Baltimore, MD, 1996.
2. S. F. ASHBY, T. A. MANTEUFFEL AND P. E. SAYLOR, *A taxonomy for conjugate gradient methods*, SIAM J. Numer. Anal., 27 (1990), pp. 1542-1568.
3. D. P. OLEARY, *The block conjugate gradient algorithm and related methods*, Linear Algebra Appl., 29 (1980), pp. 293322.
4. M. ROBBÉ AND M. SADKANE, *Exact and inexact breakdowns in the block GMRES method*, Linear Algebra and its Applications, 419:265-285, 2006.
5. M. R. HESTENES AND E. STIEFEL, *Methods of conjugate gradients for solving linear systems*, Journal of research of the National Bureau of Standards, 49:409-436, 1952.
6. A. BHAYA, P. BLIMAN, G. NIEDU AND F. A. PAZOS, *A cooperative conjugate gradient method for linear systems permitting multithread implementation of low complexity*, Proceedings of the 51th IEEE Conference on Decision and Control, CDC 2012, Maui, HI, USA. 638-643, December 2012.
7. 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 Analysis and Applications, 16(4):1135-1153, 1995.
8. L. GRIGORI, S. MOUFAWAD AND F. NATAF, *Enlarged Krylov Subspace Conjugate Gradient Methods for Reducing Communication*, INRIA Technical Report 8597, submitted to SIAM J. Matrix Analysis and Applications, 2016.
9. L. GRIGORI, F. NATAF AND S. YOUSEF, *Robust algebraic Schur complement preconditioners based on low rank corrections*, INRIA TR 8557, 2014.
10. M .H. GUTKNECHT, *Block Krylov space methods for linear systems with multiple right-hand sides: an introduction*, In: A. SIDDIQI, I. DUFF AND O. CHRISTENSEN (eds.), Modern Mathematical Models, Methods and Algorithms for Real World Systems, pp. 420-447. Anamaya Publishers, New Delhi (2007).
11. B. LOWERYAND AND J. LANGOU, *Stability Analysis of QR factorization in an Oblique Inner Product*, ArXive-prints, January 2014.
12. A. A. DUBRULLE, *Retooling the method of block conjugate gradients*, ETNA, Electronic Transactions on Numerical Analysis [electronic only] 12 (2001): 216-233.
13. C. BREZINSKI, *Multiparameter descent methods*, Linear Algebra Appl., 296(1-3):113141, 1999.
14. C. BREZINSKI AND F. BANTEGNIES, *The multiparameter conjugate gradient algorithm*, Matapli, 75 (2004) 67–85.
15. J. LANGOU, *Iterative methods for solving linear systems with multiple right-hand sides*, Ph.D. dissertation, TH/PA/03/24, CERFACS, France, 2003.
16. A. CHAPMAN AND Y. SAAD, *Deflated and augmented Krylov subspace techniques*, Numer. Linear Algebra Appl., 4(1):4366, 1997.
17. A. GAUL, M. H. GUTKNECHT, J. LIESEN AND R. NABBEN, *A framework for deflated and augmented Krylov subspace methods*, SIAM J. Matrix Anal. Appl., 34(2):495518, 2013.
18. N. SPILLANE, *An Adaptive Multi Preconditioned Conjugate Gradient Algorithm*, Submitted, Preprint available online hal-01170059.
19. R. BRIDSON AND C. GREIF, *A multipreconditioned conjugate gradient algorithm*, SIAM J. Matrix Anal. Appl., 27(4):10561068 (electronic), 2006.
20. V. FABER AND T. A. MANTEUFFEL, *Necessary and sufficient conditions for the existence of a conjugate gradient method*, SIAM J. Numer. Anal., 21 (1984), pp. 352-362.
21. Y. ACHDOU, F. NATAF, *Low frequency tangential filtering decomposition*, Numerical Linear Algebra with Applications, 14(2):129147, 2007.
22. Q. NIU, L. GRIGORI, P. KUMAR AND F. NATAF, *Modified tangential frequency filtering decomposition and its Fourier analysis*, Numerische Mathematik, 116(1):123148, 2010.

DOI: 10.1002/nla

23. F. NATAF, F. HECHT, P. JOLIVET, AND C. PRUDHOMME, *Scalable Domain Decomposition Preconditioners For Heterogeneous Elliptic Problems*, SC13, (Denver, Colorado, United States), p. 11 p., ACM, 2013.
24. S. F. ASHBY, M. .J. HOLST, T. .A. MANTEUFFEL AND P. .E. SAYLOR, The role of the inner product in stopping criteria for conjugate gradient iteratiosn, BIT, 41, pp. 26-53, 2001.
25. M. ROZLOZNIK, J. KOPAL, M. TUMA AND A. SMOKTUNOWICZ, *Numerical stability of orthogonalization methods with a non-standard inner product*, BIT Numerical Mathematics 52, pp. 1035-1058, 2012.
26. Y. SAAD AND M. H. SCHULTZ, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 856-869.
27. Y. SAAD, *Iterative methods for sparse linear systems*, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, second edition, 2003.
28. R. FEZZANI, L. GRIGORI, F. NATAF, AND K. WANG, /em Block filtering decomposition, Numerical Linear Algebra with Applications Journal, Volume 21, Issue 6, pages 703 - 721, 2014
29. F. HECHT, *New development in freefem++*, J. Numer. Math., vol. 20, no. 3-4, pp. 251265, 2012.
30. J. W. DEMMEL, L. GRIGORI, M. HOEMMEN, AND J. LANGOU, *Communication-optimal parallel and sequential QR and LU factorizations*, SIAM Journal on Scientific Computing, (2012), pp. 206–239. short version of technical report UCB/EECS-2008-89 from 2008.

DOI: 10.1002/nla