SIAM J. MATRIX ANAL. APPL.
Vol. 37, No. 2, pp. 744–773

ENLARGED KRYLOV SUBSPACE CONJUGATE GRADIENT METHODS FOR REDUCING COMMUNICATION*

LAURA GRIGORI†, SOPHIE MOUFAWAD‡, AND FREDERIC NATAF†

Abstract. In this paper we introduce a new approach for reducing communication in Krylov subspace methods that consists of enlarging the Krylov subspace by a maximum of t vectors per iteration, based on a domain decomposition of the graph of A. The obtained enlarged Krylov subspace $\mathcal{K}_{k,t}(A, r_0)$ is a superset of the Krylov subspace $\mathcal{K}_k(A, r_0)$, $\mathcal{K}_k(A, r_0) \subset \mathcal{K}_{k,t}(A, r_0)$. Thus, we search for the solution of the system $Ax = b$ in $\mathcal{K}_{k,t}(A, r_0)$ instead of $\mathcal{K}_k(A, r_0)$. Moreover, we show in this paper that the enlarged Krylov projection subspace methods lead to faster convergence in terms of iterations and parallelizable algorithms with less communication, with respect to Krylov methods.

Key words. minimizing communication, linear algebra, iterative methods

AMS subject classifications. 65F10, 68W10

DOI. 10.1137/140989492

1. Introduction.

Krylov subspace methods are among the most practical and popular iterative methods today. They are polynomial iterative methods that aim to solve systems of linear equations ($Ax = b$) by finding a sequence of vectors $x_1, x_2, x_3, x_4, \ldots, x_k$ that minimizes some measure of error over the corresponding spaces

$$x_0 + \mathcal{K}_i(A, r_0), \quad i = 1, \ldots, k,$$

where $\mathcal{K}_i(A, r_0) = span\{r_0, Ar_0, A^2r_0, \ldots, A^{i-1}r_0\}$ is the Krylov subspace of dimension i, x_0 is the initial iterate, and r_0 is the initial residual. Conjugate gradient (CG) [18], generalized minimal residual (GMRES) [29], bi-conjugate gradient [21, 8], and bi-conjugate gradient stabilized [31] are some of the most used Krylov subspace methods.

These methods are governed by Blas1 and Blas2 operations as dot products and sparse matrix vector multiplications. Parallelizing dot products is constrained by communication since the performed computation is negligible. If the dot products are performed by one processor, then there is a need for a communication before and after the computation. In both cases, communication is a bottleneck. This problem has been tackled by different approaches. First, block methods that solve systems with multiple right-hand sides $AX = B$ were introduced, as block CG (B-CG) [25]. Then, s-step methods that compute s basis vectors per iteration were proposed, examples are s-step CG [32, 4] and s-step GMRES [33, 7]. Both methods, block and s-step, use Blas2 and Blas3 operations. Recently, communication avoiding methods, based on s-step methods, that aim at avoiding communication at the expense of performing some redundant flops were introduced, as CA-CG, CA-GMRES [23, 19], and CA-ILU0 preconditioner [12]. Another approach is to hide the cost of communication

*Received by the editors October 13, 2014; accepted for publication (in revised form) by A. Frommer March 30, 2016; published electronically June 21, 2016.
http://www.siam.org/journals/simax/37-2/98949.html
†CNRS UMR 7598, Laboratoire Jacques-Louis Lions Paris, INRIA Paris-Rocquencourt, Alpines, and UPMC - Univ Paris 6, Paris 75005, France (laura.grigori@inria.fr, nataf@ann.jussieu.fr).
‡Corresponding author. CNRS UMR 7598, Laboratoire Jacques-Louis Lions Paris, INRIA Paris-Rocquencourt, Alpines, and UPMC - Univ Paris 6, France (smm33@aub.edu.lb).

by overlapping it with other computation, like pipelined CG [6, 15] and pipelined GMRES [9].

In this paper we introduce a new approach that consists of enlarging the Krylov subspace by a maximum of t vectors per iteration. First, the input matrix is partitioned into t subdomains by using a graph partitioning algorithm. At the beginning of the iterative method, the residual is split into t vectors corresponding to the t subdomains. Then, the obtained t vectors are multiplied by A at each iteration to generate t new basis vectors. The obtained enlarged Krylov subspace $\mathcal{K}_{k,t}(A, r_0)$ is a superset of the Krylov subspace $\mathcal{K}_k(A, r_0)$, $\mathcal{K}_k(A, r_0) \subset \mathcal{K}_{k,t}(A, r_0)$. Thus it is possible to search for the solution of the system $Ax = b$ in $\mathcal{K}_{k,t}(A, r_0)$ instead of $\mathcal{K}_k(A, r_0)$. Moreover, we show in this paper that the enlarged Krylov projection subspace methods lead to faster convergence in terms of iterations and parallelizable algorithms with less communication, with respect to Krylov methods.

The enlarged Krylov subspace methods can be considered as a special case of the augmented Krylov subspace methods [3, 28] since, in some sense, we are augmenting the classical subspace. However, the subtle difference is that we are not adding some subspace spanned by other vectors, like the eigenvalues in deflation methods [3]. But we are taking the same classical Krylov subspace and enlarging it by splitting each vector into t vectors. Moreover, the enlarged Krylov subspace methods should not be confused with block Krylov methods, a special case of augmented Krylov methods, that solve a system with multiple right-hand sides. The enlarged Krylov subspace methods solve one system with one residual vector at each iteration, but with multiple basis vectors or multiple search directions obtained from the decomposition of the domain.

In this paper we focus on CG [18], a Krylov projection method for symmetric (Hermitian) positive definite matrices (SPD), which was introduced by Hestenes and Stiefel in 1952 (section 2.1). After giving a brief overview of related existing CG methods (section 2) such as B-CG [25], cooperative CG (coop-CG) [1], and multiple search direction CG (MSD-CG) [16], we discuss several new versions of CG (section 3). The first method, multiple search direction with orthogonalization CG (MSDO-CG), is an adapted version of MSD-CG [16]. MSD-CG has the same structure as the classical CG method where first t new search directions are defined on the t subdomains, then the t step lengths are obtained by solving a $t \times t$ system, and finally the solution and the residual are updated. But unlike CG, the search directions are not A-orthogonal. In MSDO-CG, the search directions are A-orthonormalized to obtain a projection method that guarantees convergence at least as fast as CG. The idea of using more than one search direction was also exploited in Rixen's thesis [26] for two subdomains in the context of domain decomposition methods, and further developed in [11].

The second method that we propose here, long recurrence enlarged CG (LRE-CG), is similar to GMRES in that we build an orthonormal basis for the enlarged Krylov subspace rather than finding search directions. Then, we use the whole basis to update the solution and the residual. We show that this method is a projection method and hence should converge at least as fast as CG. The third set of introduced methods, short recurrence enlarged CG (SRE-CG), SRE-CG2, and truncated SRE-CG2, have the short recurrence property, as their name indicates. These methods build an A-orthonormal basis rather than an orthonormal basis, as in LRE-CG. The difference between the three methods is in the A-orthonormalization process, where the first A-orthonormalizes the t computed basis vectors against the previous $2t$ vectors, the second A-orthonormalizes the t computed basis vectors against all the previous vectors, and the third A-orthonormalizes the t computed basis vectors against a subset of the previous vectors. We compare the convergence behavior of all

the introduced methods using different A-orthonormalization and orthonormalization methods and then we compare the most stable versions with CG and other related methods (section 4).

We test our methods on matrices arising from the dicretization of two-dimensional (2D) Poisson equations (Poisson2D), three-dimensional (3D) elasctisity equations (Elasticity3D), and 2D and 3D convection-diffusion equations such as Nh2D, Sky2D, Sky3D, and Ani3D as discussed in section 4. All the methods converge faster than CG in terms of iterations, but LRE-CG and SRE-CG2 converge faster than MSDO-CG, SRE-CG, and truncated SRE-CG2. And the more subdomains are introduced or the larger t is, the faster is the convergence of the enlarged methods with respect to CG in terms of iterations. For example, for $t = 64$, MSDO-CG, LRE-CG, and SRE-CG2 methods converge in 75% to 89% fewer iterations than CG for the matrices Nh2D, Poisson2D, and Elasticity3D, and 95% to 98% fewer iterations than CG for the matrices Sky2D, Sky3D, and Ani3D. But increasing t also means increasing the memory requirements for the methods MSDO-CG, LRE-CG, and SRE-CG2. Thus, in practice, t should be relatively small, depending on the available memory, on the size of the matrix, and on the number of iterations needed for convergence, as explained in section 4. However, the memory requirements of SRE-CG and truncated SRE-CG2 are fixed and unrelated to t. We briefly discuss the parallel versions of the introduced algorithms along with their expected performance based on the estimated run times in section 5.

2. Overview of existing CG methods. The Krylov projection methods find a sequence of approximate solutions x_k $(k > 0)$ of the system $Ax = b$, and are defined by the following two conditions:

1. Subspace condition: $x_k \in x_0 + \mathcal{K}_k(A, r_0)$.
2. Petrov–Galerkin condition: $r_k \perp \mathcal{L}_k \iff (r_k)^t y = 0 \ \ \forall \ \ y \in \mathcal{L}_k$,

where $\mathcal{K}_k(A, r_0) = span\{r_0, Ar_0, A^2r_0, \ldots, A^{k-1}r_0\}$ is the Krylov subspace of dimension k, x_0 is the initial iterate, r_0 is the initial residual, and $\mathcal{L}_k$ is a well-defined subspace of dimension k. The classical CG is a Krylov projection method, where $\mathcal{L}_k = \mathcal{K}_k(A, r_0)$.

In this section we briefly introduce the CG versions related to our MSDO-CG, LRE-CG, SRE-CG, and SRE-CG2 versions, starting with the 1952 Hestenes and Stiefel version (section 2.1). Since then, many differenet versions of CG have been introduced (refer to [10] for a historical overview of CG till 1976). In 1980 O'Leary introduced a B-CG version [25] that solves a system with multiple right-hand sides $AX = B$ (section 2.2). coop-CG [1] which was recently introduced, solves the system $Ax = b$ by starting with t distinct initial guesses. MSD-CG [16] solves $Ax = b$ by decomposing A's domain into t subdomains and defining a search direction on each of the t subdomains. Unlike CG, B-CG, and coop-CG, MSD-CG does not have the A-orthogonality condition of the search directions. Hence it is not a projection method.

Note that in this paper we use MATLAB notation for matrices and vectors. For example, given a vector p of size $n \times 1$ and a set of indices δ, $p(\delta)$ is the vector formed by the subset of the entries of p whose indices belong to δ. For a matrix A, $A(\delta, :)$ is a submatrix formed by the subset of the rows of A whose indices belong to δ. Similarly, $A(:, \alpha)$, is a submatrix formed by the subset of the columns of A whose indices belong to α. And $A(\alpha, \beta) = [A(\alpha, :)](:, \beta)$, is formed by the β columns of the submatrix $A(\alpha, :)$.

2.1. CG method. CG [18] is an iterative Krylov projection method for SPD matrices of the form

$$Ax = b. \tag{2.1}$$

Given an initial guess or iterate x_0, at the kth iteration CG finds the new approximate solution $x_k = x_{k-1} + \alpha_k p_k$ that minimizes $\phi(x) = \frac{1}{2}(x)^t Ax - b^t x$ over the corresponding space $x_0 + \mathcal{K}_k(A, r_0)$, where $k > 0$, $p_k \in \mathcal{K}_k(A, r_0)$ is the kth search direction, and α_k is the step along the search direction.

The minimum of $\phi(x)$ is given by $\nabla\phi(x) = 0$, which is equivalent to $\nabla\phi(x) = Ax - b = 0$. Thus, by minimizing $\phi(x)$ we are solving the system (2.1). As the name of the method indicates, the gradients $\nabla\phi(x_i)$ for all i should be conjugate. And since CG is a Krylov projection method, the residual $r_k = b - Ax_k$ should respect the Petrov–Galerkin condition

$$r_k \perp \mathcal{L}_k,$$

where r_k is orthogonal to some well-defined subspace $\mathcal{L}_k \subseteq \mathbb{R}^n$ (or $\subseteq \mathbb{C}^n$) of dimension k. In CG, the subspace $\mathcal{L}_k$ is the same as the Krylov subspace $\mathcal{K}_k$. Thus, $r_k^t y = 0$ for all $y \in \mathcal{K}_k$. Hence, the residuals form an orthogonal set, $r_k^t r_i = 0$, for all $i < k$.

Moreover, the Petrov–Galerkin condition $r_k \perp \mathcal{K}_k(A, r_0)$ is equivalent to the conjugacy of the gradients $\nabla\phi(x_k)^t \nabla\phi(x_i) = 0$, for all $i \neq k$. Once x_k has been chosen, either x_k is the required approximate solution of $Ax = b$ or a new search direction $p_{k+1} \neq 0$ must be determined to compute the new approximation $x_{k+1} = x_k + \alpha_{k+1} p_{k+1}$. This procedure is repeated until convergence or untill the maximum number of allowed iterations has been reached without convergence. The convergence criterion is set as

$$||r_k||_2 \leq \epsilon ||b||_2 \text{ for some } \epsilon \in \mathbb{R},$$

where $r_k = b - Ax_k \in \mathcal{K}_{k+1}(A, r_0)$ is the kth residual.

THEOREM 2.1. *The Petrov–Galerkin condition $(r_k)^t y = 0$ for all $y \in \mathcal{K}_k$ implies the A-orthogonality of the search directions $p_i^t A p_j = 0$ for all $i \neq j$ and $i, j \leq k$.*

Proof. Refer to [13] for the proof. ∎

This theorem means that the A-orthogonality of the search directions has to be ensured or else the Petrov–Galerkin condition won't be respected. On the other hand, the search direction $p_k \in \mathcal{K}_k$ is chosen according to the following recursion relation,

$$\begin{cases} p_1 = r_0, \\ p_k = r_{k-1} + \beta_k p_{k-1}, \end{cases} \tag{2.2}$$

where p_1 is set equal to r_0 since the initial residual is equal to negative the gradient $-\nabla\phi(x_0)$ which is the steepest descent from x_0. But p_k is not set to r_{k-1}, the steepest descent from x_{k-1} for $k > 1$, since the residuals are not A-orthogonal. It can be shown that the search directions defined in (2.2) are A-orthogonal, i.e., $p_k^t A p_i = 0$ for all $i \leq k - 1$. For $i < k - 1$, we have

$$p_k^t A p_i = r_{k-1}^t A p_i + \beta_k p_{k-1}^t A p_i = \beta_k p_{k-1}^t A p_i \tag{2.3}$$

since $r_{k-1}^t A p_i = 0$ by the Petrov–Galerkin condition. In addition, $r_{k-1}^t p_i = r_{k-2}^t p_i - \alpha_{k-1} p_{k-1}^t A p_i = 0$ with $r_{k-2}^t p_i = 0$ since $i \leq k - 2$. Thus, $p_{k-1}^t A p_i = 0$. Therefore, $p_k^t A p_i = 0$ for $i < k - 1$.

As for $i = k - 1$, $r_{k-1}^t A p_{k-1} \neq 0$ and $p_{k-1}^t A p_{k-1} \neq 0$ for $p_{k-1} \neq 0$. Thus, $\beta_k = -\frac{(r_{k-1})^t A p_{k-1}}{(p_{k-1})^t A p_{k-1}}$ is chosen so that $p_k^t A p_{k-1} = 0$.

At each iteration, the step $\alpha_k = \frac{(p_k)^t r_{k-1}}{(p_k)^t A p_k} = \frac{||r_{k-1}||_2^2}{||p_k||_A^2}$ is chosen such that

$$\phi(x_k) = \min\{\phi(x_{k-1} + \alpha p_k), \forall \alpha \in \mathbb{R}\}.$$

Using the definition of α_k, then $\beta_k = -\frac{(r_{k-1})^t A p_{k-1}}{(p_{k-1})^t A p_{k-1}} = \frac{||r_{k-1}||_2^2}{||r_{k-2}||_2^2}$.

2.2. B-CG method. In 1980 O'Leary introduced a B-CG version [25] that solves an SPD system with multiple right-hand sides

$$AX = B, \tag{2.4}$$

where A is an $n \times n$ matrix, $X \in \mathbb{R}^{n\times t}$ is a block vector, and B is a block vector of size $n \times t$ containing the multiple right-hand sides.

Starting with an initial guess $X_0 \in \mathbb{R}^{n\times t}$, initial residual $R_0 = B - AX_0$, $P_1 = R_0\gamma_1$ with γ_1 a $t \times t$ full rank freely chosen matrix, the B-CG searches for an approximate solution $X_{k+1} \in X_0 + \mathcal{K}_{k+1}(A, R_0)$, where $\mathcal{K}_{k+1}(A, R_0) = block - span\{R_0, AR_0, A^2R_0, \ldots, A^kR_0\}$ is the block Krylov subspace. By the Petrov–Galerkin condition we have that $R_{k+1} \perp \mathcal{K}_{k+1}(A, R_0)$. Then, $R_{k+1}^t Y = 0$ for all $Y \in \mathcal{K}_{k+1}(A, R_0)$, where $Y = \sum_{i=1}^{k} A^i R_0 \zeta_i$ and ζ_i is a $t \times t$ matrix. This implies that $R_{k+1}^t R_i = 0$ and $R_{k+1}^t AP_i = 0$ for all $i < k + 1$.

Then, for $k \geq 0$ the iterates are defined similarly to CG:

$$\begin{aligned} X_k &= X_{k-1} + P_k\alpha_k && \in \mathcal{K}_k(A, R_0), \\ R_k &= R_{k-1} - AP_k\alpha_k && \in \mathcal{K}_{k+1}(A, R_0), \\ P_{k+1} &= (R_k + P_k\beta_{k+1})\gamma_{k+1} && \in \mathcal{K}_{k+1}(A, R_0), \end{aligned}$$

where

$$\begin{aligned} \alpha_k &= (P_k^t AP_k)^{-1}\gamma_k^t(R_{k-1}^t R_{k-1}), \\ \beta_{k+1} &= \gamma_k^{-1}(R_{k-1}^t R_{k-1})^{-1}(R_k R_k). \end{aligned}$$

Note that α_k is chosen such that $\phi(X_k) = \min\{\phi(X_{k-1} + P_k\alpha)\ \forall\alpha \in \mathbb{R}^{t,t}\}$. As for β_k, it is chosen to ensure the A-orthogonality of the P_k and P_{k-1} ($(P_{k-1})^t AP_k = 0$), whereas γ_k is a $t \times t$ full rank matrix that can be chosen freely to decrease roundoff errors in the implementation. Moreover, the search direction $P_{k+1} \in \mathcal{K}_{k+1}(A, R_0)$ of the B-CG method is A-conjugate, $(P_{k+1})^t AY = 0$ for all $Y \in \mathcal{K}_k(A, R_0)$. This leads to the A-orthogonality of the search direction $\Longrightarrow (P_{k+1})^t AP_i = 0$ for all $i < k + 1$.

2.3. coop-CG method. Recently, in 2012, Bhaya et al. presented a parallel version of CG for solving $Ax = b$, which is based on the B-CG method. The idea of using block methods for solving a system with one right-hand side is not new. For example, in [3] Chapman and Saad proposed the use of block GMRES for improving the convergence of a system with one right-hand side, by defining the block residual R_0 as $r_0 = b - Ax_0$ and $t - 1$ random vectors such as approximate eigenvectors. However, the coop-CG [1] solves the system $Ax = b$ by starting with t different initial guesses and solving the same system t times in parallel, where t threads/agents cooperate to find the solution. This is equivalent to solving the system $AX = b * \mathbb{1}_t$, where X_0 is a block vector containing the t initial guesses, $R_0 = AX_0 - b * \mathbb{1}_t$ is the block residual, $P_1 = R_0$ is the initial block search direction, and $\mathbb{1}_t$ is a vector of ones of size $1 \times t$. Then the derivations and the algorithm of the coop-CG are the same as the B-CG with $\gamma_k = I$, $\rho_k = \min(||R_k(:, 1)||_2^2, ||R_k(:, 2)||_2^2, \ldots, ||R_k(:, t - 1)||_2^2, ||R_k(:, t)||_2^2)$, and stopping criteria $\sqrt{\rho_k} > \epsilon||b||_2$ and $k < k_{\max}$. Once the method has converged, the solution is the ith column of X_k corresponding to the ith column of R_k with the minimum norm. This method has faster convergence than CG (section 2.3).

2.4. MSD-CG method. The MSD-CG method, introduced by Gu et al. [16], solves the system $Ax = b$, and starts by having a decomposed domain and by defining at each iteration k a search direction $p_{k,i}$ on each of the t subdomains (δ_i, $i = 1, 2, \ldots, t$) such that $p_{k,i}(\delta_j) = 0$ for all $j \neq i$. Then, the approximate solution at the kth iteration is defined as $x_k = x_{k-1} + P_k\alpha_k$, where $P_k = [p_{k,1}\ p_{k,2}\ p_{k,3}\ \ldots\ p_{k,t}]$

is a matrix containing all the kth search directions and α_k is a vector of size t:

$$P_k = \begin{pmatrix} * & 0 & & 0 & 0 \\ \vdots & \vdots & & \vdots & \vdots \\ * & 0 & & 0 & 0 \\ 0 & * & & 0 & 0 \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & * & & 0 & 0 \\ & & \ddots & & \\ 0 & 0 & & * & 0 \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & & * & 0 \\ 0 & 0 & & 0 & * \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & & 0 & * \end{pmatrix}_{n \times t}. \tag{2.5}$$

Given an initial guess x_0, the residual is defined as $r_k = b - Ax_k$ for $k \geq 0$. The first set of domain search directions is defined by the initial residual r_0, such that $p_{1,i}(\delta_i) = r_0(\delta_i)$ for $i = 1, 2, \ldots, t$ and zero otherwise. Then, for $k > 1$ the domain search directions are defined as follows: $p_{k,i} = T_i(r_{k-1}) + \beta_{k,i} p_{k-1,i}$ for $i = 1, 2, \ldots, t$, where $\beta_{k,i}$ is a scalar and T_i is an operator that projects a vector onto the subdomain δ_i ($[T_i(x)](\delta_j) = 0$ for $j \neq i$ and $[T_i(x)](\delta_i) = x(\delta_i)$). The sparsity pattern of the search directions block for all k is shown in (2.5).

As for $\alpha_k = (P_k^t A P_k)^{-1} P_k^t r_{k-1}$, it is chosen such that it minimizes $\phi(x_k) = \min\{\phi(x_{k-1} + P_k \alpha), \forall \alpha \in \mathbb{R}^t\}$. Unlike CG, B-CG, and coop-CG, MSD-CG does not have the A-orthogonality condition of the search directions, i.e., $P_k^t A P_i$ is not equal to zero for all i not equal to k. Thus $\beta_k = (P_{k-1}^t A P_{k-1})^{-1} P_{k-1}^t A r_{k-1}$ is chosen so that the global search direction $p_k = \sum_{i=1}^t p_{k,i}$ is A-orthogonal to the previous domain search direction $p_{k-1,i}$, i.e., $(p_k)^t A P_{k-1} = 0$ for $i = 1, 2, \ldots, t$. As for the convergence, it is shown that the rate of convergence of MSD-CG is at least as fast as that of the steepest descent method. Yet, steepest descent is known for its slow "zig-zagging" convergence. This causes the MSD-CG to have slower convergence than CG, and in some cases it does not converge at all with respect to the given stopping criteria as shown in section 4.

3. The new CGs. We introduce several new CG methods, MSDO-CG, LRE-CG, SRE-CG, and SRE-CG2, which are based on replacing the Krylov subspace $\mathcal{K}_k$ with a larger subspace, specifically the enlarged Krylov subspace, leading to better convergence. Thus we will first introduce the new enlarged Krylov subspace and its properties in the context of CG methods in section 3.1. Then in sections 3.2 and 3.3 we introduce the MSDO-CG, the LRE-CG, the SRE-CG, and a second variant, SRE-CG2.

As previously mentioned, MSDO-CG is an adapted version of MSD-CG, where the t newly defined search directions are A-orthonormalized against previous search directions and against each other. This A-orthonormalization guarantees a convergence behavior at least as good as CG. In [2], the authors introduce a similar method referred to as multiple preconditioned CG (MPCG). They also choose to maintain the A-orthogonality of the search directions, as we do in MSDO-CG. However, they define the multiple search directions by using multiple preconditioners. The methods are closely related with the difference that MSDO-CG can be preconditioned, whereas MPCG does have an unpreconditioned version.

In the case of LRE-CG, the enlarged Krylov subspace is formed by computing at each iteration t new basis vectors. Rather than having short recurrences, x_k is defined

by all the basis vectors as in GMRES, where the basis vectors are orthonormalized. Both methods, MSDO-CG and LRE-CG, require saving at most tk vectors versus one search direction in CG. Yet LRE-CG converges faster than MSDO-CG (section 4) at the expense of solving growing systems of size tk. Several remedies to this problem are discussed in [13].

Similarly to LRE-CG, SRE-CG computes t new basis vectors at each iteration k. But, as the name indicates, we obtain a short recurrence version, since the t computed basis vectors are A-orthonormalized against the previous $2t$ vectors. This version requires saving only $3t$ vectors. However, it is possible to lose the A-orthogonality of the whole basis in finite arithmetics. Thus, in SRE-CG2, the t computed basis vectors are A-orthonormalized against all the previous basis vectors. But we end up storing tk basis vectors. A remedy to this problem would be to have a truncated A-orthonormalization SRE-CG2 version, where the t new basis vectors are A-orthonormalized against the previous ti vectors, and $3 \leq i \leq k-2$ is chosen based on the available memory.

3.1. The enlarged Krylov subspace. The enlarged Krylov subspace and methods are based on a partition of the unknowns or, alternatively, the rows of the $n \times n$ matrix A. Assume that the index domain $\delta = \{1, 2, \ldots, n\}$ is divided into t distinct subdomains δ_i, where $\delta = \cup_{i=1}^{t} \delta_i$.

We define $T_i(x)$ to be the operator that projects the vector x onto the subdomain δ_i. Let $y = T_i(x)$, then $y(\delta_i) = x(\delta_i)$ and zero elsewhere. Then, we define $T(x)$ to be an operator that transforms the $n \times 1$ vector x into t vectors of size $n \times 1$ that correspond to the projection of x onto the subdomains δ_i for $i = 1, 2, \ldots, t$. Moreover, we define $\mathcal{T}(x)$ to be an operator that transforms the $n \times 1$ vector x into an $n \times t$ matrix containing the t vectors obtained from $T(x)$. Thus, $\mathcal{T}(x)$ is different from $T(x)$ since $\mathcal{T}(x)$ is a matrix, whereas $T(x) = \{T_1(x), T_2(x), \ldots, T_t(x)\}$ is a set of vectors. But $\mathcal{T}(x) = [T_1(x)\, T_2(x)\, \ldots .\, T_t(x)] = [T(x)]$, where the brackets $[\ldots]$ denote a matrix format.

DEFINITION 3.1. *Let*

$$\begin{aligned}\mathcal{K}_{k,t} &= span\{T(r_0), AT(r_0), A^2T(r0), \ldots, A^{k-1}T(r_0)\} \\ &= span\{T_1(r_0), T_2(r_0), \ldots, T_t(r_0), AT_1(r_0), AT_2(r_0), \ldots, AT_t(r_0), \ldots, \\ &\qquad A^{k-1}T_1(r_0), \ldots, A^{k-1}T_t(r_0)\}\end{aligned}$$

be an enlarged Krylov subspace of dimension $k \leq z \leq tk$ generated by the matrix A and the vector r_0, and associated with a given partition defined by δ_i for $i = 1, 2, \ldots, t$.

The enlarged Krylov subspaces $\mathcal{K}_{k,t}(A, r_0)$ are increasing subspaces, yet bounded. We denote by k_u the upper bound k for which the dimension of the enlarged Krylov subspace $\mathcal{K}_{k,t}(A, r_0)$ stops increasing. For simplicity, we will denote the enlarged Krylov subspace generated by A and r_0, $\mathcal{K}_{k,t}(A, r_0)$, by $\mathcal{K}_{k,t}$, and the Krylov subspace generated by A and r_0, $\mathcal{K}_k(A, r_0)$ by $\mathcal{K}_k$.

THEOREM 3.2. *The Krylov subspace $\mathcal{K}_k$ is a subset of the enlarged Krylov subspace $\mathcal{K}_{k,t}$ ($\mathcal{K}_k \subset \mathcal{K}_{k,t}$).*

Proof. Let $y \in \mathcal{K}_k$, where $\mathcal{K}_k = span\{r_0, Ar_0, \ldots, A^{k-1}r_0\}$. Then

$$y = \sum_{j=0}^{k-1} a_j A^j r_0 = \sum_{j=0}^{k-1} a_j A^j \mathcal{T}(r_0) * \mathbb{1}_t = \sum_{j=0}^{k-1} \sum_{i=1}^{t} a_j A^j T_i(r_0) \in \mathcal{K}_{k,t}$$

since $r_0 = \mathcal{T}(r_0) * \mathbb{1}_t = [T_1(r_0)\, T_2(r_0)\, \ldots\, T_t(r_0)] * \mathbb{1}_t$. ∎

Krylov subspace methods search for an approximate solution $x_k \in x_0 + \mathcal{K}_k$. A corollary of Theorem 3.2 is that we can search for an approximate solution x_k in $x_0 + \mathcal{K}_{k,t}$ instead, since $\mathcal{K}_k \subset \mathcal{K}_{k,t}$.

In Theorem 3.3, we do not use the direct sum $\oplus$ since it is not guaranteed that the intersection of the two subspaces, $\mathcal{K}_{k,t}$ and $span\{A^kT_1(r_0), A^kT_2(r_0), \ldots, A^kT_t(r_0)\}$, is empty.

THEOREM 3.3. *By Definition* 3.1 *of the enlarged Krylov subspace,*

$$\mathcal{K}_{k+1,t} = \mathcal{K}_{k,t} + span\{A^kT_1(r_0), A^kT_2(r_0), \ldots, A^kT_t(r_0)\}.$$

If $A^kT_v(r_0) \in \mathcal{K}_{k,t}$ *for all* $1 \le v \le t$, *then* $A^{k+q}T_i(r_0) \in \mathcal{K}_{k,t}$ *for some* $1 \le i \le t$ *and for some* $q > 0$.

Proof. We prove this by induction. Refer to [13]. ∎

Given that $\mathcal{K}_{k,t} \neq \mathcal{K}_{k-1,t}$ and $A^kT_v(r_0) \in \mathcal{K}_{k,t}$ for all $1 \le v \le t$, then a corollary of Theorem 3.3 is that $\mathcal{K}_{k,t} = \mathcal{K}_{k+q,t}$ for all $q > 0$, and $k_u = k$ is the upper bound for which the dimension of the enlarged Krylov subspace stops increasing.

THEOREM 3.4. *If*

$$A^kT_i(r_0) \in \mathcal{K}_{k,t} + span\{A^kT_1(r_0), \ldots, A^kT_{i-1}(r_0), A^kT_{i+1}(r_0), \ldots, A^kT_t(r_0)\},$$

then

$$A^{k+q}T_i(r_0) \in \mathcal{K}_{k+q,t} + span\{A^{k+q}T_1(r_0), \ldots, A^{k+q}T_{i-1}(r_0), A^{k+q}T_{i+1}(r_0), \ldots, A^{k+q}T_t(r_0)\}$$

for all $1 \le i \le t$ *and* $q > 0$.

Proof. If $A^kT_i(r_0) \in \mathcal{K}_{k,t} + span\{A^kT_1(r_0), \ldots, A^kT_{i-1}(r_0), A^kT_{i+1}(r_0), \ldots, A^kT_t(r_0)\}$, then $A^kT_i(r_0) = \sum_{u=0}^{k-1}\sum_{v=1}^{t}\alpha_{u,v}A^uT_v(r_0) + \sum_{\substack{v=1\\v\neq i}}^{t}\alpha_{k,v}A^kT_v(r_0)$. Thus,

$$\begin{aligned} A^{k+q}T_i(r_0) &= \sum_{u=0}^{k-1}\sum_{v=1}^{t}\alpha_{u,v}A^{u+q}T_v(r_0) + \sum_{\substack{v=1\\v\neq i}}^{t}\alpha_{j,v}A^{k+q}T_v(r_0) \\ &\in \mathcal{K}_{k+q,t} + span\{A^{k+q}T_1(r_0), \ldots, A^{k+q}T_{i-1}(r_0), \\ &\qquad A^{k+q}T_{i+1}(r_0), \ldots, A^{k+q}T_t(r_0)\} \end{aligned}$$

∎

A corollary of Theorem 3.4 is that if $t - i_k$ vectors of the form $A^kT_y(r_0)$ with $y = i_k + 1, \ldots, t$ belong to the subspace $\mathcal{K}_{k,t} + span\{A^kT_1(r_0), A^kT_2(r_0), \ldots, A^kT_{i_k}(r_0)\}$, then the $t - i_k$ vectors of the form $A^{k+q}T_y(r_0)$ belong to the subspace

$$\mathcal{K}_{k+q,t} + span\{A^{k+q}T_1(r_0), A^{k+q}T_2(r_0), \ldots, A^{k+q}T_{i_k}(r_0)\}.$$

THEOREM 3.5. *Let* k_u *be the smallest integer such that* $\mathcal{K}_{k_u,t} = \mathcal{K}_{k_u+q,t}$ *for all* $q > 0$. *Then, for all* $k < k_u$ *the dimension of the enlarged Krylov subspaces* $\mathcal{K}_{k,t}$ *and* $\mathcal{K}_{k+1,t}$ *is strictly increasing by some number* i_k *and* i_{k+1}, *respectively, where* $1 \le i_{k+1} \le i_k \le t$.

Proof. By the definition of k_u, we have that for all $q > 0$

$$\mathcal{K}_{1,t} \subsetneq \cdots \subsetneq \mathcal{K}_{k_u-1,t} \subsetneq \mathcal{K}_{k_u,t} = \mathcal{K}_{k_u+q,t}.$$

Then for all $k < k_u$, the dimension of the enlarged Krylov subspaces $\mathcal{K}_{k,t}$ is strictly increasing by some number $i_k \neq 0$ with respect to the dimension of $\mathcal{K}_{k-1,t}$.

In general, $dim(\mathcal{K}_{k,t}) = dim(\mathcal{K}_{k-1,t}) + i_k$, where $1 \le i_k \le t$ and $dim()$ is the dimension of a subspace. Similarly, $dim(\mathcal{K}_{k+1,t}) = dim(\mathcal{K}_{k,t}) + i_{k+1}$, where $1 \le i_{k+1} \le t$. Moreover, in $\mathcal{K}_{k,t}$'s basis we added i_k new vectors of the form $A^{k-1}T_i(r_0)$,

while the other $t - i_k$ either belong to $\mathcal{K}_{k-1,t}$ or are linearly dependant on the i_k vectors and $\mathcal{K}_{k-1,t}$. In both cases, the $t - i_k$ vectors of the form $A^{k-1}T_i(r_0)$ belong to the subspace $\mathcal{K}_{k-1,t} + span\{A^{k-1}T_1(r_0), \ldots, A^{k-1}T_{i_k}(r_0)\}$. Then by Theorem 3.4 and its corollary, the $t - i_k$ vectors of the form $A^{k+q}T_i(r_0)$ belong to the subspace $\mathcal{K}_{k+q,t}+span\{A^{k+q}T_1(r_0), A^{k+q}T_2(r_0), \ldots, A^{k+q}T_{i_k}(r_0)\}$ for $q > 0$.Therefore, we have at least $t - i_k$ linearly dependent vectors added to $\mathcal{K}_{k+1,t}$, hence i_{k+1} can never be greater than i_k. ∎

THEOREM 3.6. *Let p_u and k_u be such that $\mathcal{K}_{p_u} = \mathcal{K}_{p_u+q}$ and $\mathcal{K}_{k_u,t} = \mathcal{K}_{k_u+q,t}$ for $q > 0$. Then $k_u \leq p_u$.*

Proof. Let $\mathcal{K}_{p_u} = \mathcal{K}_{p_u+q}$ and $A^{p_u+q-1}r_0 \in \mathcal{K}_{p_u+q}$, where $q > 0$. Then $A^{p_u+q-1}r_0 \in \mathcal{K}_{p_u} \subset \mathcal{K}_{p_u,t}$, and $A^{p_u+q-1}r_0 = \sum_{j=1}^{p_u}\sum_{i=1}^{t}\alpha_{j,i}A^{j-1}T_i(r_0)$. Thus $A^{p_u+q-1}\sum_{i=1}^{t}T_i(r_0)$ $= \sum_{j=1}^{p_u}\sum_{i=1}^{t}\alpha_{j,i}A^{j-1}T_i(r_0)$.

Suppose that $A^{p_u+q-1}T_i(r_0) \notin \mathcal{K}_{p_u,t}$, for all $1 \leq i \leq t$. Then $A^{p_u+q-1}\sum_{i=1}^{t}T_i(r_0)$ $= \sum_{j=1}^{p_u+q-1}\sum_{i=1}^{t}\alpha_{j,i}A^{j-1}T_i(r_0)$. We may assume that there exists at least one $\alpha_{j,i} \neq 0$ for $j > p_u$, then this leads to a contradiction. This implies that $A^{p_u+q-1}T_i(r_0) \in \mathcal{K}_{p_u,t}$ for all $1 \leq i \leq t$.

Thus by definition of the $T()$ operator and since $\mathcal{K}_p$ is a subset of $\mathcal{K}_{p,t}$, if $\mathcal{K}_{p_u} = \mathcal{K}_{p_u+q}$, then $\mathcal{K}_{p_u,t} = \mathcal{K}_{p_u+q,t}$. However, if $\mathcal{K}_{k_u,t} = \mathcal{K}_{k_u+q,t}$ this does not imply that $\mathcal{K}_{k_u} = \mathcal{K}_{k_u+q}$. Therefore, since $\mathcal{K}_{k,t}$ is a much larger subspace than $\mathcal{K}_k$, it is possible to reach stagnation earlier. Therefore $k_u \leq p_u$. ∎

THEOREM 3.7. *The solution of the system $Ax = b$ belongs to the subspace $x_0 + \mathcal{K}_{k_u,t}$, where $\mathcal{K}_{k_u+q,t} = \mathcal{K}_{k_u,t}$ for $q > 0$.*

Proof. The solution $x_{sol} \in x_0 + \mathcal{K}_{p_u}$, where $\mathcal{K}_{p_u} = span\{r_0, Ar_0, \ldots, A^{p_u-1}r_0\}$ and $\mathcal{K}_{p_u} = \mathcal{K}_{p_u+q}$ for $q > 0$. Since $\mathcal{K}_{p_u} \subset \mathcal{K}_{p_u,t}$, the solution $x_{sol} \in x_0 + \mathcal{K}_{p_u,t}$, where $p_u \geq k_u$ by Theorem 3.6.

Suppose that $x_{sol} \in x_0 + \mathcal{K}_{p_u,t}$, but $x_{sol} \notin x_0 + \mathcal{K}_{k_u,t}$. This implies that $\mathcal{K}_{k_u,t} \neq \mathcal{K}_{p_u,t}$. However, by the definition of k_u and since $k_u \leq p_u$, we have that $\mathcal{K}_{k_u,t} = \mathcal{K}_{p_u,t}$. This is a contradiction. ∎

3.1.1. Krylov projection methods. The Krylov projection methods find a sequence of approximate solutions x_k $(k > 0)$ of the system $Ax = b$ from the subspace $x_0 + \mathcal{K}_k \subseteq \mathbb{R}^n$ (or $\subseteq \mathbb{C}^n$) by imposing the Petrov–Galerkin constraint on the kth residual $r_k = b - Ax_k$, that is, r_k is orthogonal to some well-defined subspace of dimension k.

We define our new enlarged Krylov projection methods based on CG by the subspace $\mathcal{K}_{k,t}$ and the following two conditions:

1. Subspace condition: $x_k \in x_0 + \mathcal{K}_{k,t}$.
2. Orthogonality condition: $r_k \perp \mathcal{K}_{k,t} \iff (r_k)^t y = 0$ for all $y \in \mathcal{K}_{k,t}$,

where $\mathcal{K}_{k,t}$ is a well-defined subspace of dimension $k \leq z \leq tk$.

3.1.2. The minimization property. The new enlarged CG methods find the new approximate solution by minimizing the function $\phi(x)$ over the subspace $x_0+\mathcal{K}_{k,t}$.

THEOREM 3.8. *If $r_k \perp \mathcal{K}_{k,t}$, then $\phi(x_k) = \min\{\phi(x), \forall x \in x_0 + \mathcal{K}_{k,t}\}$.*

Proof. By the orthogonality condition we have that $r_k \perp \mathcal{K}_{k,t}$

$$\begin{aligned} \implies (r_k)^t y &= 0 \qquad \forall y \in \mathcal{K}_{k,t}, \\ (b - Ax_k)^t y &= 0 \qquad \forall y \in \mathcal{K}_{k,t}, \\ b^t y - (x_k)^t A y &= 0 \qquad \forall y \in \mathcal{K}_{k,t}. \end{aligned}$$

Then, for all $x \in x_0 + \mathcal{K}_{k,t}$, we have

$$\begin{aligned}
\phi(x) - \phi(x_k) &= \frac{1}{2}x^tAx - b^tx - \left[\frac{1}{2}(x_k)^tAx_k - b^tx_k\right] \\
&= \frac{1}{2}x^tAx - b^t(x - x_k) - \frac{1}{2}(x_k)^tAx_k, \text{ where } (x - x_k) \in \mathcal{K}_{k,t} \\
&= \frac{1}{2}x^tAx - (x_k)^tA(x - x_k) - \frac{1}{2}(x_k)^tAx_k, \text{ since } b^t(x - x_k) \\
&= (x_k)^tA(x - x_k) \\
&= \frac{1}{2}x^tAx - (x_k)^tAx + \frac{1}{2}(x_k)^tAx_k \\
&= \frac{1}{2}(x - x_k)^tA(x - x_k) \geq 0, \text{ since } A \text{ is positive definite.}
\end{aligned}$$

Thus, $\phi(x) \geq \phi(x_k)$ for all $x \in x_0 + \mathcal{K}_{k,t}$. ∎

THEOREM 3.9. *$\phi(x_k) = \min\{\phi(x), \forall x \in x_0 + \mathcal{K}_{k,t}\}$ if and only if $||x^* - x_k||_A = \min\{||x^* - x||_A, \forall x \in x_0 + \mathcal{K}_{k,t}\}$, where x^* is the exact solution of* (2.1).

Proof. $f(x) = ||x^* - x||_A = (x^*)^tAx^* - 2(x^*)^tAx + x^tAx = b^tx^* - 2b^tx + x^tAx = b^tx^* + 2\phi(x)$. The minimum of $f(x)$ is given by $f'(x) = \nabla\phi(x) = 0$. ∎

3.1.3. Convergence analysis. The CG method of Hestenes and Stiefel is known to converge in $\overline{K}$ iterations, where $\overline{K} \leq n$, if the matrix $A \in \mathbb{R}^{n,n}$ is SPD. Moreover, the kth error of CG $\overline{e}_k = ||x^* - \overline{x}_k|| \leq 2(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1})^k||\overline{e}_0||_A$, where $\kappa = ||A||_2||A^{-1}||_2 = \frac{\lambda_{\max}}{\lambda_{\min}}$ is the L2-condition number of the SPD matrix A, $\lambda_{\max}$ is the largest eigenvalue of A, and $\lambda_{\min}$ is the smallest eigenvalue of A .

Assuming that the kth residual of the new CG methods satisfies the orthogonality condition, $r_k \perp \mathcal{K}_{k,t}$, then by Theorems 3.8 and 3.9 we have that

$$\begin{aligned}
||e_k||_A = ||x^* - x_k||_A &= \min\{||x^* - x||_A, \forall x \in x_0 + \mathcal{K}_{k,t}\} \\
&\leq \min\{||x^* - \overline{x}||_A, \forall \overline{x} \in x_0 + \mathcal{K}_k\} \text{ since } \mathcal{K}_k \subset \mathcal{K}_{k,t} \\
&\leq ||\overline{e}_k||_A.
\end{aligned}$$

Therefore, our methods converge at least as fast as the classical CG method, assuming that the orthogonality condition ($r_k \perp \mathcal{K}_{k,t}$) is respected. Hence, the enlarged Krylov subspace CG methods will converge in K iterations, where $K \leq \overline{K} \leq n$.

3.2. MSDO-CG method. The MSD-CG method introduced by Gu et al. [16] can be viewed as an enlarged Krylov method, where $P_0 = \mathcal{T}(r_0)$, and $P_k = \mathcal{T}(r_{k-1}) + P_{k-1}diag(\beta_k)$ for $i = 1, 2, \ldots, t$, $x_k = x_{k-1} + P_k\alpha_k$, and $r_k = r_{k-1} - AP_k\alpha_k$ with $\alpha_k = (P_k^tAP_k)^{-1}P_k^tr_{k-1}$ and $\beta_k = (P_{k-1}^tAP_{k-1})^{-1}P_{k-1}^tAr_{k-1}$. However, the P_k's are not A-othogonal implying that $r_k \not\perp \mathcal{K}_{k,t}$. Thus, MSD-CG is not a projection method.

MSDO-CG is an enlarged Krylov projection method that solves the system (2.1) (Ax=b), by approximating the solution at the kth iteration with the vector $x_k = x_{k-1} + P_k\alpha_k$ such that

$$\phi(x_k) = \min\{\phi(x), \forall x \in \mathcal{K}_{k,t}\},$$

where $P_k\alpha_k \in \mathcal{K}_{k,t}$, P_k is an $n \times t$ block vector containing the t subdomain search directions, and α_k is a vector of size t.

The minimum of $\phi(x)$ is given by $\nabla\phi(x) = 0$, which is equivalent to $Ax - b = 0$. Thus, by minimizing $\phi(x)$, we are solving the system (2.1). Note that since $\phi(x_k) = \min\{\phi(x), \forall x \in x_0 + \mathcal{K}_{k,t}\}$, then

$$\phi(x_k) = \phi(x_{k-1} + P_k\alpha_k) = \min\{\phi(x_{k-1} + P_k\alpha), \forall\alpha \in \mathbb{R}^t\}. \tag{3.1}$$

Once x_k has been chosen, either x_k is the desired solution of $Ax = b$, or t new domain search direction vectors P_{k+1} and a new approximation $x_{k+1} = x_k + P_{k+1}\alpha_{k+1}$ are computed. Similarly to MSD-CG, we choose to define $P_{k+1} = [p_1^{k+1}\ p_2^{k+1}\ \dots\ p_t^{k+1}]$, where $p_i^1 = T_i(r_0)$ and $p_i^{k+1} = T_i(r_k) + \beta_i^{k+1}p_i^k$ for $i = 1, 2, \dots, t$. But unlike MSD-CG, MSDO-CG is a projection method. Hence, we A-orthonormalize all the search directions, P_{k+1}, to ensure that $r_{k+1} \perp \mathcal{K}_{k+1,t}$ as discussed in section 3.2.2. By imposing the orthogonality condition, $r_{k+1} \perp \mathcal{K}_{k+1,t}$, it is guaranteed that MSDO-CG converges at least as fast as CG as proven in section 3.1.3. This procedure is repeated until convergence. Thus, we need to find the recursion relations of r_k, P_k, $\alpha_k = [\alpha_1^k, \alpha_2^k, \dots, \alpha_t^k]^t$, and $\beta_k = [\beta_1^k, \beta_2^k, \dots, \beta_t^k]^t$.

3.2.1. The residual r_k. By definition, the residual $r_k = b - Ax_k$, where $x_k \in \mathcal{K}_{k,t}$. Thus $r_k \in \mathcal{K}_{k+1,t}$. As for the recursion relation of r_k, we simply replace x_k by its expression and obtain the following:

$$\begin{aligned} r_k &= b - Ax_k \\ &= b - A(x_{k-1} + P_k\alpha_k) \\ &= r_{k-1} - AP_k\alpha_k. \end{aligned}$$

Moreover, if the orthogonality condition, $r_k \perp \mathcal{K}_{k,t}$, is ensured, then $(r_k)^t r_i = 0$ for all $i < k$. Hence, the residuals form an orthogonal set.

THEOREM 3.10. *The orthogonality condition $(r_k)^t y = 0$ for all $y \in \mathcal{K}_{k,t}$ implies the A-orthogonality of the block search directions $P_i^t AP_j = 0$, for all $i \neq j$ and $i, j \leq k$.*

Proof. By definition, the column vectors of P_i belong to $\mathcal{K}_{i,t}$ and $\mathcal{K}_{i,t} \subset \mathcal{K}_{i+1,t}$. Thus, the column vectors of P_i belong to $\mathcal{K}_{i+q,t}$ for $q \geq 0$. By the orthogonality condition $r_{k-1}^t P_i = 0$ for $i \leq k-1$ and $r_k^t P_i = 0$. Thus, $r_k^t P_i = r_{k-1}^t P_i - \alpha_k^t P_k^t AP_i = 0$ for $i \leq k-1$. This implies that $P_k^t AP_i = 0$ for $i \leq k-1$ since $\alpha_k \neq 0$ and therefore, the A-orthogonality of the search directions. ∎

3.2.2. The domain search direction P_k. Similarly to MSD-CG, we choose to define the domain search direction as

$$P_k = \mathcal{T}(r_{k-1}) + P_{k-1}diag(\beta_k), \tag{3.2}$$

where $diag(\beta_k)$ is a $t \times t$ matrix with the vector β_k on the diagonal.

Another option would be to define the search directions as

$$P_k = \mathcal{T}(r_{k-1}) + P_{k-1}\beta_k, \tag{3.3}$$

where β_k is a $t \times t$ matrix.

In both cases, the domain search directions defined in (3.2) and (3.3) are not A-orthogonal to each other. To ensure that the orthogonality condition is valid, at each iteration k the block vector P_k is A-orthonormalized against all the previous P_i, where $i = 1, 2, \dots, k-1$. Then the column vectors of P_k are A-orthonormalized against each other. Thus, the obtained search directions $\widetilde{P}_k$ satisfy $(\widetilde{P}_k)^t A\widetilde{P}_i = 0$ for all $i \neq k$.

Moreover, $(\widetilde{P}_k)^t A\widetilde{P}_k = I$, where I is the identity matrix, assuming that the column vectors of P_k are linearly independant with respect to each other and the previous directions or, alternatively, none of the column vectors of $\widetilde{P}_k$ are zero. Note that, once $P_k = \mathcal{T}(r_{k-1}) + P_{k-1}diag(\beta_k)$ is defined, it is directly A-orthonormalized. Thus, in the sections that follow, we denote by P_k the A-orthonormalized search directions and we do not use the $\widetilde{P}_k$ notation to be consistent with the initial definitions in the previous sections.

There are several A-orthonormalization methods. First, for A-orthonormalizing P_k against all the previous P_i, where $i = 1, 2, \ldots, k-1$, one can use classical Gram–Schmidt (CGS), modified Gram–Schmidt (MGS), or classical Gram–Schmidt with reorthogonalization (CGS2), where the CGS algorithm is applied twice for numerical stability reasons. As for A-orthonormalizing P_k, there are many methods that are discussed in [22, 27], but we will only refer to CGS, CGS2, MGS, A-CholQR, and Pre-CholQR. We seek a combination of both A-orthonormalizations that is stable and parallelizable with reduced communication. For that reason, in section 4 we test the MSDO-CG method with the different combinations of the A-orthonormalization methods and we conclude that the MSDO-CG is numerically most stable when we use MGS+MGS, CGS2+A-CholQR, or CGS2+Pre-CholQR. In section 5, we discuss the parallelization of the MSDO-CG algorithm with the stable A-orthonormalization methods.

3.2.3. Finding the expressions of α_k and β_k. At each iteration, the step α_k is chosen such that $\phi(x_k) = \min\{\phi(x_{k-1} + P_k\alpha), \forall\alpha \in \mathbb{R}^t\}$.

Let $F(\alpha) = \phi(x_{k-1} + P_k\alpha)$, where $\phi(x) = \frac{1}{2}x^tAx - x^tb$. Then,

$$\begin{aligned}
F(\alpha) &= \frac{1}{2}(x_{k-1} + P_k\alpha)^tA(x_{k-1} + P_k\alpha) - (x_{k-1} + P_k\alpha)^tb \\
&= \phi(x_{k-1}) + \frac{1}{2}[(x_{k-1})^tAP_k\alpha + \alpha^t(P_k)^tAx_{k-1} + \alpha^t(P_k)^tAP_k\alpha] - \alpha^t(P_k)^tb \\
&= \phi(x_{k-1}) + \frac{1}{2}[(x_{k-1})^tAP_k\alpha - \alpha^t(P_k)^tAx_{k-1}] + \frac{1}{2}\alpha^t(P_k)^tAP_k\alpha - \alpha^t(P_k)^tr_{k-1} \\
&= \phi(x_{k-1}) + \frac{1}{2}\alpha^t(P_k)^tAP_k\alpha - \alpha^t(P_k)^tr_{k-1},
\end{aligned}$$

since A is SPD.

The minimum of $F(\alpha)$ is given by $F'(\alpha) = 0$

$$\Rightarrow F'(\alpha) = (P_k)^tAP_k\alpha - (P_k)^tr_{k-1} = 0.$$

Therefore, $\alpha_k = (P_k^tAP_k)^{-1}(P_k^tr_{k-1})$.

As for β_k, it should be chosen to ensure that P_k is A-orthogonal to P_{k-1}. $P_k = \mathcal{T}(r_{k-1}) + P_{k-1}diag(\beta_k)$ and $P_{k-1}^tAP_k = P_{k-1}^tA\mathcal{T}(r_{k-1}) + P_{k-1}^tAP_{k-1}diag(\beta_k)$. Since P_{k-1} is an A-orthonormal matrix, $P_{k-1}^tAP_{k-1} = I$, $diag(\beta_k)$ should be equal to $-P_{k-1}^tA\mathcal{T}(r_{k-1})$. But nothing guarantees that $P_{k-1}^tA\mathcal{T}(r_{k-1})$ is a diagonal matrix. So we choose $\beta_k = -(P_{k-1}^tAP_{k-1})^{-1}P_{k-1}^tAr_{k-1}$ which guarantees that $P_k * \mathbb{1}_t$ is A-orthogonal to P_{k-1}, similarly to MSD-CG. Moreover, in case $P_{k-1}^tA\mathcal{T}(r_{k-1})$ is a diagonal matrix, then our choice of β_k implies that P_k is A-orthogonal to P_{k-1}. If $t = 1$, then MSDO-CG is reduced to the classical CG. Note that in case Definition (3.3) is used to define the search directions, then $\beta_k = -(P_{k-1}^tAP_{k-1})^{-1}P_{k-1}^tA\mathcal{T}(r_{k-1})$ is chosen so that P_k is A-orthogonal to P_{k-1}.

Since the vectors of P_k are A-orthonormalized ($P_k^tAP_k = I$), then α_k and β_k systems are reduced to $\alpha_k = P_k^tr_{k-1}$ and $\beta_k = -P_{k-1}^tAr_{k-1}$. These are the main algorithmic differences with MSD-CG [16] (section 2.4).

3.2.4. The MSDO-CG algorithm. After deriving the recurrence relations of x_k, r_k, P_k, α_k, and β_k, we present the MSDO-CG algorithm in Algorithm 1. We do not specify the A-orthonormalization methods, since this choice will be based first on the numerical stability of the method (section 4), then on its parallelization with the least communication possible (section 5).

Algorithm 1 MSDO-CG algorithm	Flops
Input: A, the $n \times n$ SPD matrix	
Input: b, the $n \times 1$ right-hand side; x_0, the initial guess or iterate	
Input: ϵ, the stopping tolerance; $k_{\max}$, the maximum allowed iterations	
Output: x_k, the approximate solution of the system $Ax = b$	
1: $r_0 = b - Ax_0$, $\rho = \|\|r_0\|\|_2^2$, $k = 1$	$2\text{nnz} + 2n - 1$
2: Let $P_1 = \mathcal{T}(r_0)$ and $W_1 = AP_1$	$2\text{nnz} - (t-1)n$
3: **while** $(\sqrt{\rho} > \epsilon\|\|b\|\|_2$ and $k < k_{\max})$ **do**	$2n$
4: **if** k==1 **then**	
5: A-orthonormalize P_1 and update W_1	not included here
6: **else**	
7: $\beta_k = -(P_{k-1}^t W_{k-1})^{-1}(W_{k-1}^t r_{k-1}) = -W_{k-1}^t r_{k-1}$	$(2n-1)t$
8: $P_k = \mathcal{T}(r_{k-1}) + P_{k-1} diag(\beta_k)$	$2nt$
9: $W_k = A\mathcal{T}(r_k) + W_k diag(\beta_k)$	$2\text{nnz}-(t-1)n+2nt$
10: A-orthonormalize P_k against all P_i's and update W_k	not included here
11: A-orthonormalize P_k and update W_k	not included here
12: **end if**	
13: $\alpha_k = (P_k^t W_k)^{-1}(P_k^t r_{k-1}) = P_k^t r_{k-1}$	$(2n-1)t$
14: $x_k = x_{k-1} + P_k \alpha_k$	$(2t-1)n + n$
15: $r_k = r_{k-1} - W_k \alpha_k$	$(2t-1)n + n$
16: $\rho = \|\|r_k\|\|_2^2$	$2n-1$
17: $k = k + 1$	1
18: **end while**	

Thus we present the MSDO-CG algorithm (Algorithm 1) and the computed flops per iteration except for the A-orthonormalization steps. To reduce communication and computation in the A-orthonormalization steps, be it MGS, CGS, CGS2, A-CholQR, or Pre-CholQR, we replace $W_k = AP_k$ by

$$\begin{cases} W_1 = AP_1, \\ W_k = A\mathcal{T}(r_{k-1}) + AP_{k-1} diag(\beta_k) \quad \forall k > 1, \\ \quad\; = A\mathcal{T}(r_{k-1}) + W_{k-1} diag(\beta_k). \end{cases}$$

This is discussed in further detail in the technical report [13], which this article is based on, specifically in Algorithms 14, 15, 18, 21, 25, and 27. Then, the cost of A-orthonormalizing P_k against previous vectors using MGS, CGS, or CGS2 methods is $(6n-1)t^2k+4nt$ flops, $(6n-1)t^2k+3nt$ flops, or $(12n-2)t^2k+6nt$ flops, respectively. And the cost of A-orthonormalizing P_k using MGS, A-CholQR, or Pre-CholQR is $(6n-1)\frac{t^2}{2} + nt$ flops, $4nt^2 + 4nt$ flops, or $4\text{nnz}t + 5nt^2 - nt$ flops, respectively.

The total number of flops computed sequentially in Algorithm 1 after k iterations, except for the A-orthonormalizations, is

$$\begin{aligned} \text{Total Flops} &= 4\text{nnz} - nt + 5n - 1 + k(11nt - 2t + 2n - 1 + 2\text{nnz} + n + 1) \\ &= 4\text{nnz} - nt + 5n - 1 + k(11nt + 3n - 2t + 2\text{nnz}) \\ &\approx 4\text{nnz} + 5n + k(11nt + 2\text{nnz}), \end{aligned}$$

which is of the order of nnz$k + ntk$ flops, where nnz is the number of nonzero entries in the $n \times n$ matrix A and t is the number of search directions computed at each iteration.

It must be noted that since the P_i's are A-orthonormal to each others, then the $t \times t$ matrix $P_k^t W_k = P_k^t A P_k$ is the identity matrix. Hence, solving for α_k and β_k requires computing matrix vector multiplications.

3.3. SRE-CG method. In this section, we introduce a class of enlarged Krylov projection CG methods that solves the system $Ax = b$ by approximating the solution at the kth iteration with the vector $x_k = x_{k-1} + Q_k \alpha_k \in x_0 + \mathcal{K}_{t,k}$ such that

$$\phi(x_k) = \min\{\phi(x), \forall x \in x_0 + \mathcal{K}_{t,k}\},$$

where $Q_k \alpha_k \in \mathcal{K}_{t,k}$ and Q_k is an $n \times tk$ matrix containing the basis vectors of $\mathcal{K}_{t,k}$ and $\phi(x) = \frac{1}{2} x^t A x - x^t b$. We present three versions that differ in the way the basis is constructed. However, the general derivations are the same.

As mentioned earlier, the minimum of $\phi(x)$ is given by $\bigtriangledown \phi(x) = 0$ which is equivalent to $Ax - b = 0$. Thus, by minimizing $\phi(x)$ we are solving the system $Ax = b$. Since $\phi(x_k) = \min\{\phi(x), \forall x \in x_0 + \mathcal{K}_{t,k}\}$, then

$$\phi(x_k) = \phi(x_{k-1} + Q_k \alpha_k) = \min\{\phi(x_{k-1} + Q_k \alpha), \forall \alpha \in \mathbb{R}^{tk}\}. \tag{3.4}$$

Once x_k has been chosen, either x_k is the exact solution of $Ax = b$, or t new basis vectors and the new approximation $x_{k+1} = x_k + Q_{k+1} \alpha_{k+1}$ are computed. This procedure is repeated until convergence.

Thus, we need to find the recursion relations of r_k and α_k. By definition, the residual $r_k = b - A x_k$, where $x_k \in x_0 + \mathcal{K}_{t,k}$. Thus $r_k \in \mathcal{K}_{t,k+1}$. The recursion relation of r_k can be simply obtained by replacing x_k by its expression as follows:

$$\begin{aligned} r_k &= b - A x_k \\ &= b - A(x_{k-1} + Q_k \alpha_k) \\ &= r_{k-1} - A Q_k \alpha_k. \end{aligned}$$

At each iteration the step α_k is chosen such that

$$\phi(x_k) = \min\{\phi(x_{k-1} + Q_k \alpha), \forall \alpha \in \mathbb{R}^{t(k+1)}\}.$$

Let $F(\alpha) = \phi(x_{k-1} + Q_k \alpha)$, where $\phi(x) = \frac{1}{2} x^t A x - x^t b$. Then,

$$\begin{aligned} F(\alpha) &= \frac{1}{2}(x_{k-1} + Q_k \alpha)^t A (x_{k-1} + Q_k \alpha) - (x_{k-1} + Q_k \alpha)^t b \\ &= \phi(x_{k-1}) + \frac{1}{2}[(x_{k-1})^t A Q_k \alpha + \alpha^t (Q_k)^t A x_{k-1} + \alpha^t (Q_k)^t A Q_k \alpha] - \alpha^t (Q_k)^t b \\ &= \phi(x_{k-1}) + \frac{1}{2}[(x_{k-1})^t A Q_k \alpha - \alpha^t (Q_k)^t A x_{k-1}] + \frac{1}{2}\alpha^t (Q_k)^t A Q_k \alpha - \alpha^t (Q_k)^t r_{k-1} \\ &= \phi(x_{k-1}) + \frac{1}{2}\alpha^t (Q_k)^t A Q_k \alpha - \alpha^t (Q_k)^t r_{k-1}, \quad \textit{since } A \textit{ is SPD}. \end{aligned}$$

The minimum of $F(\alpha)$ is given by $F'(\alpha) = 0$

$$\Rightarrow F'(\alpha) = (Q_k)^t A Q_k \alpha - (Q_k)^t r_{k-1} = 0.$$

Therefore, $\alpha_k = (Q_k^t A Q_k)^{-1} (Q_k^t r_{k-1})$.

By minimizing $\phi(x)$, the orthogonality condition, $r_k \perp \mathcal{K}_{t,k}$, is ensured (Theorem 3.11). Therefore, $(r_k)^t r_i = 0$ for all $i < k$, and the residuals form an orthogonal set. Then at each iteration k we have

$$\begin{aligned} x_k &= x_{k-1} + Q_k \alpha_k, \\ r_k &= r_{k-1} - AQ_k \alpha_k, \\ \alpha_k &= (Q_k^t A Q_k)^{-1} (Q_k^t r_{k-1}). \end{aligned}$$

THEOREM 3.11. *Assuming that $x_k = x_{k-1} + Q_k \alpha_k$, then the orthogonality condition, $r_k \perp \mathcal{K}_{t,k}$, is equivalent to x_k being the minimum of $\phi(x)$ in $x_0 + \mathcal{K}_{t,k}$.*

Proof.

1. x_k is the minimum of $\phi(x)$ in $x_0 + \mathcal{K}_{t,k}$ implies $r_k \perp \mathcal{K}_{t,k}$. The minimum of $F(\alpha) = \phi(x_k) = \phi(x_{k-1} + Q_k \alpha)$ is given by $F'(\alpha) = (Q_k)^t A Q_k \alpha - (Q_k)^t r_{k-1} = 0$. Since x_k is the minimum, then $\alpha = \alpha_k$ and $F'(\alpha) = -Q_k^t r_k = 0$. Thus $r_k \perp \mathcal{K}_{t,k}$.
2. $r_k \perp \mathcal{K}_{t,k}$ implies x_k is the minimum of $\phi(x)$ in $x_0 + \mathcal{K}_{t,k}$. Proof by contradiction: Assume that $r_k \perp \mathcal{K}_{t,k}$ and x_k is not the minimum of $\phi(x)$ in $x_0 + \mathcal{K}_{t,k}$. Then $F'(\alpha_k) \neq 0$. Hence $Q_k^t r_k \neq 0$ and r_k is not orthogonal to $\mathcal{K}_{t,k}$. This contradicts our assumption. Thus x_k is the minimum of $\phi(x)$. ∎

The monomial basis vectors of $\mathcal{K}_{t,k}$ are $\{T(r_0), AT(r_0), \ldots A^{k-1}T(r_0)\}$. We can either orthonormalize or A-orthonormalize the basis. In case we orthonormalize the basis vectors, then we obtain a long recurrence enlarged CG version, that is expensive in terms of flops since we have to solve, at each iteration k, the system $\alpha_k = (Q_k^t A Q_k)^{-1}(Q_k^t r_{k-1})$ of size $tk \times tk$, where Q_k is the matrix containing the set of orthonormal basis vectors of $\mathcal{K}_{t,k}$. For a detailed description of the LRE-CG algorithm refer to [13].

Another alternative is to A-orthonormalize the basis vectors rather than orthonormalizing them. Then we obtain the following. First, $\alpha_k = (Q_k^t A Q_k)^{-1}(Q_k^t r_{k-1}) = Q_k^t r_{k-1}$ since Q_k is an A-orthonormal basis, i.e., $Q_k^t A Q_k = I$. Moreover, by the orthogonality condition, $Q_{k-1}^t r_{k-1} = 0$. Thus, $Q_k^t r_{k-1} = [Q_{k-1}\ W_k]^t r_{k-1} = [0_{t(k-1)}; W_k^t r_{k-1}]$, where W_k is the set of t newly computed vectors, and α_k is a $tk \times 1$ vector. Hence, $\alpha_k = [0_{t(k-1)}; \tilde{\alpha}_k]$, where $\tilde{\alpha}_k = W_k^t r_{k-1}$.

Then,

$$\begin{aligned} x_k &= x_{k-1} + Q_k \alpha_k \\ &= x_{k-1} + [Q_{k-1} W_k][0_{t(k-1)}; \tilde{\alpha}_k] \\ &= x_{k-1} + W_k \tilde{\alpha}_k, \end{aligned}$$

where $\tilde{\alpha}_k = W_k^t r_{k-1}$. Similarly, $r_k = r_{k-1} - AW_k \tilde{\alpha}_k$.

Note that in exact arithmetic the A-orthonormalization of W_k against $Q_{k-1} = [W_1\ W_2\ \ldots\ W_{k-1}]$ can be summarized as follows:

$$\begin{aligned} W_k &= AW_{k-1} - Q_{k-1} Q_{k-1}^t A(AW_{k-1}) \\ &= AW_{k-1} - \sum_{i=1}^{k-1} W_i W_i^t A(AW_{k-1}) \\ &= AW_{k-1} - W_{k-1} W_{k-1}^t A(AW_{k-1}) - W_{k-2} W_{k-2}^t A(AW_{k-1}), \end{aligned}$$

since $(AW_i)^t AW_{k-1} = 0$ for all $i < k-2$ by the A-orthonormality of the basis vectors of $\mathcal{K}_{t,k-1}$. We call this version short recurrence enlarged CG, since unlike LRE-CG

we only need the last $3t$ computed vectors, x_{k-1} and r_{k-1}, to define x_k and r_k. The method is summarized in Algorithm 2.

However, in finite arithmetic there might be a loss of A-orthogonality between the last set of computed basis vectors and the first ones. Thus, one can A-orthonormalize W_k against all the basis vectors. We call this version SRE-CG2, where we need the last t computed vectors, x_{k-1} and r_{k-1}, to define x_k and r_k. But we still need to save all the tk basis vectors in order to A-orthonormalize W_k against Q_{k-1}. The SRE-CG2 Algorithm 3 is the same as Algorithm 2 except for line 7 where we A-orthonormalize W_k against W_i for all $1 \leq i \leq k-1$. Note that in case there isn't enough memory to store the tk vectors, it is possible to use a truncated version of the A-orthonormalization against previous vectors, where W_k is A-orthonormalized against a subset of $\{W_1, W_2, \ldots, W_{k-3}\}$ along with W_{k-1} and W_{k-2}. This truncated SRE-CG2 requires less memory than SRE-CG2 and converges faster than SRE-CG in the number of iterations.

The cost of SRE-CG Algorithm 2 and SRE-CG2 Algorithm 3, except for the A-orthonormalizations in steps 7 and 8, after k iterations is

$$\begin{aligned}\text{Total Flops} &= 4\text{nnz} + 3n - 1 + 2nt + k[(2\text{nnz} + 5n - 1)t + 2n] \\ &\approx 2\text{nnz}tk.\end{aligned}$$

As for the memory requirements, in SRE-CG Algorithm 2 we have to store the matrix A , $3t + 2$ vectors of size $n \times 1$, and a $t \times 1$ vector. Whereas, in SRE-CG2 Algorithm 3, we have to store the matrix A, $tk + 2$ vectors of size $n \times 1$, and a $t \times 1$ vector, where $k \leq k_{\max}$ is the number of computed iterations. And in the truncated SRE-CG2 algorithm, we have to store the matrix A, $tk_{trunc} + 2$ vectors of size $n \times 1$, and a $t \times 1$ vector, where k_{trunc} is a fixed number such that $2 < k_{trunc} < k \leq k_{\max}$.

Algorithm 2 SRE-CG algorithm	Flops
Input: A, the $n \times n$ SPD matrix	
Input: b, the $n \times 1$ right-hand side; x_0, the initial guess or iterate	
Input: ϵ, the stopping tolerance; $k_{\max}$, the maximum allowed iterations	
Output: x_k, the approximate solution of the system $Ax = b$	
1: $r_0 = b - Ax_0$, $\rho_0 = \|\|r_0\|\|_2^2$, k = 1	$2\text{nnz} + 2n - 1$
2: **while** ($\sqrt{\rho_{k-1}} > \epsilon\|\|b\|\|_2$ and $k < k_{\max}$) **do**	$2n$
3: **if** k==1 **then**	
4: Let $W_1 = \mathcal{T}(r_0)$, and A-orthonormalize its vectors	$2\text{nnz} - n + 2nt$
5: **else**	
6: Let $W_k = AW_{k-1}$	$(2\text{nnz} - n)t$
7: A-orthonormalize the vectors of W_k against the vectors of W_{k-1} and W_{k-2} if $k > 2$	not included here
8: A-orthonormalize the vectors of W_k	not included here
9: **end if**	
10: $\tilde{\alpha}_k = (W_k^t r_{k-1})$	$(2n-1)t$
11: $x_k = x_{k-1} + W_k\tilde{\alpha}_k$	$2tn$
12: $r_k = r_{k-1} - AW_k\tilde{\alpha}_k$	$2tn$
13: $\rho_k = \|\|r_k\|\|_2^2$	$2n - 1$
14: k = k+1	1
15: **end while**	

Algorithm 3 SRE-CG2 algorithm	Flops
Input: A, the $n \times n$ SPD matrix	
Input: b, the $n \times 1$ right-hand side; x_0, the initial guess or iterate	
Input: ϵ, the stopping tolerance; $k_{\max}$, the maximum allowed iterations	
Output: x_k, the approximate solution of the system $Ax = b$	
1: $r_0 = b - Ax_0$, $\rho_0 = \|\|r_0\|\|_2^2$, k = 1	$2\text{nnz} + 2n - 1$
2: **while** ($\sqrt{\rho_{k-1}} > \epsilon \|\|b\|\|_2$ and $k < k_{\max}$) **do**	$2n$
3: **if** k==1 **then**	
4: A-orthonormalize $W_1 = \mathcal{T}(r_0)$, and let $Q = W_1$	$2\text{nnz} - n + 2nt$
5: **else**	
6: Let $W_k = AW_{k-1}$	$(2\text{nnz} - n)t$
7: A-orthonormalize the vectors of W_k against Q not included here	
8: A-orthonormalize the vectors of W_k and let $Q = [Q\ W_k]$	not included here
9: **end if**	
10: $\tilde{\alpha} = (W_k^t r_{k-1})$	$(2n-1)t$
11: $x_k = x_{k-1} + W_k\tilde{\alpha}$	$2tn$
12: $r_k = r_{k-1} - AW_k\tilde{\alpha}$	$2tn$
13: $\rho_k = \|\|r_k\|\|_2^2$	$2n - 1$
14: k = k+1	1
15: **end while**	

4. Convergence results. After introducing the new CG methods, MSDO-CG, LRE-CG, SRE-CG, and SRE-CG2, we compare their convergence behavior with respect to different A-orthonormalization and orthonormalization schemes. Then we compare the convergence behavior of these methods with respect to CG, coop-CG, MSD-CG on several matrices for different numbers of partitions (2, 4, 8, 16, 32, and 64 partitions) or number of initial guesses (for coop-CG only). The matrices are first reordered using Metis's kway partitioning [20] that defines the subdomains δ_i. Then x is chosen randomly using the "rand" function of MATLAB and $b = Ax$. Note that the ELASTICITY3D matrix A is first scaled, due to very large values of the order of 10^{30} on the diagonal obtained from FreeFem++ [17], and then b is computed. In Tables 2, 3, and 5, "Iter" is the number of iterations, k_c, needed for convergence and "Err" is the relative error $\frac{||x-x_{k_c}||_2}{||x||_2}$ at convergence.

The first matrix POISSON2D is a block tridiagonal matrix obtained from Poisson's equation (sparse) using the MATLAB function, gallery('poisson',100). The matrices referred to as NH2D, SKY2D, SKY3D, and ANI3D, arise from boundary value problems of the convection diffusion equations

$$\begin{cases} \eta(x)u + div(\mathbf{a}(x)u) - div(\kappa(x)\nabla u) = f & in\ \Omega, \\ u = 0 & on\ \partial\Omega_D, \\ \frac{\partial u}{\partial n} = 0 & on\ \partial\Omega_N, \end{cases}$$

where $\Omega = [0,1]^n$, ($n = 2$ or 3) and $\partial\Omega_N = \partial\Omega \setminus \partial\Omega_D$. The function η, the vector field $\mathbf{a}$, and the tensor κ are the given coefficients of the partial differential operator. In the 2D case, we have $\partial\Omega_D = [0,1] \times \{0,1\}$, and in the 3D case, we have $\partial\Omega_D = [0,1] \times \{0,1\} \times [0,1]$. We focus on the following cases:

- NH2D: A nonhomogeneous problem with large jumps in the coefficients. The coefficient η and $\mathbf{a}$ are both zero. The tensor κ is isotropic and discontinuous.

TABLE 1
The test matrices.

Matrix	Size	Nonzeros	2D/3D	Problem
POISSON2D	10000	49600	2D	Poisson equations
NH2D	10000	49600	2D	Boundary value
SKY2D	10000	49600	2D	Boundary value
SKY3D	8000	53600	3D	Skyscraper
ANI3D	8000	53600	3D	Anisotropic layers
ELASTICITY3D	11253	373647	3D	Linear elasticity P1 FE

It jumps from the constant value 10^3 in the ring $\frac{1}{2\sqrt{2}} \leq |x-c| \leq \frac{1}{2}$, $c = (\frac{1}{2}, \frac{1}{2})^T$, to 1 outside.

- SKY2D and SKY3D skyscraper problems: The tensor κ is isotropic and discontinuous. The domain contains many zones of high permeability which are isolated from each other:

$$\kappa(x) = \begin{cases} 10^3 * ([10 * x_2] + 1) & \text{if } [10x_i] \text{ is odd, } i = 1, 2, \\ 1 & \text{otherwise,} \end{cases}$$

where we note $[x]$ as the integer value of x. SKY2D and SKY3D are discretized on 2D and 3D cartisian grids, respectively.

- ANI3D anisotropic layers: the domain is made of 10 anisotropic layers with jumps of up to four orders of magnitude and an anisotropy ratio of up to 10^3 in each layer. The domain is divided into 10 layers parallel to $z = 0$, of size 0.1, in which the coefficients are constant. We have $\kappa_y = 10\kappa_x$ and $\kappa_z = 1000\kappa_x$. The velocity field is zero.

POISSON2D, NH2D, and SKY2D are discretized on a 100×100 2D Cartesian grid. SKY3D and ANI3D are discretized on a $20 \times 20 \times 20$ grid.

As for the ELASTICITY3D matrix, it arises from the linear elasticity problem with Dirichlet and Neumann boundary conditions, defined as follows:

$$\begin{cases} div(\sigma(u)) + f = 0 & \text{on } \Omega, \\ u = u_D & \text{on } \partial\Omega_D, \\ \sigma(u).n = g & \text{on } \partial\Omega_N, \end{cases}$$

where Ω is a 3D $30 \times 10 \times 10$ parallelepiped, Ω_D is the Dirichlet boundary, Ω_N is the Neumann boundary, u is the unknown displacement field, f is some body force, $\sigma(u)$ is the Cauchy stress tensor given by Hooke's law. The ELASTICITY3D matrix was discretized with P1 finite elements and a triangular mesh using FreeFem++ [17]. For a detailed description of the problem refer to [14]. Table 1 briefly describes the test matrices.

In Table 2 we compare the convergence behavior of the MSDO-CG method (Algorithm 1) with different A-orthonormalization schemes for A-orthonormalizing P_k against previous P_i's (MGS, CGS, CGS2) and then A-orthonormalizing P_k against itself (MGS, CGS, CGS2, A-CholQR, Pre-CholQR) and for different numbers of partitions $t = 2, 4, 8, 16, 32, 64$ that correspond to the maximum number of vectors added at each iteration to the enlarged Krylov subspace, $\mathcal{K}_{k,t}$. We have tested different combinations of A-orthonormalizations, but we only show MGS (MGS+MGS), CGS+A-CholQR, CGS+Pre-CholQR, CGS2+A-CholQR, and CGS2+Pre-CholQR.

TABLE 2

Comparison of the convergence of MSDO-CG with different A-orthonormalization schemes, with respect to number of partitions (t) with $x_0 = 0$ and maximum iterations $k_{\max} = 2000$. The $-$ character indicates that the method did not converge in $k_{\max}$ iterations.

		MSDO-CG with different A-orthonormalization methods									
		MGS+ MGS		CGS+ A-CholQR		CGS+ Pre-CholQR		CGS2+ A-CholQR		CGS2+ Pre-CholQR	
	t	Iter	Err	Iter	Err	Iter	Err	Iter	Err	Iter	Err
POISSON2D $tol = 10^{-6}$	2	200	4E-5	204	3E-5	204	3E-5	204	3E-5	204	3E-5
	4	167	2E-5	167	2E-5	167	2E-5	167	2E-5	167	2E-5
	8	139	1E-5	139	1E-5	139	1E-5	139	1E-5	139	1E-5
	16	121	5E-6	121	5E-6	121	5E-6	121	5E-6	121	5E-6
	32	94	2E-6	94	2E-6	94	2E-6	94	2E-6	94	2E-6
	64	69	2E-6	69	2E-6	69	2E-6	69	2E-6	69	2E-6
NH2D $tol = 10^{-8}$	2	256	1E-7	256	1E-7	256	1E-7	256	1E-7	256	1E-7
	4	208	1E-7	208	1E-7	208	1E-7	208	1E-7	208	1E-7
	8	169	8E-8	169	8E-8	169	8E-8	169	8E-8	169	8E-8
	16	138	6E-8	138	6E-8	138	6E-8	138	6E-8	138	6E-8
	32	107	2E-8	107	2E-8	107	2E-8	107	2E-8	107	2E-8
	64	77	1E-8	77	1E-8	77	1E-8	77	1E-8	77	1E-8
SKY2D $tol = 10^{-8}$	2	1559	8E-4	–	–	–	–	1562	8E-4	1559	9E-4
	4	917	4E-4	–	–	–	–	917	4E-4	917	4E-4
	8	532	3E-4	–	–	–	–	531	2E-4	534	2E-4
	16	307	1E-4	–	–	–	–	307	1E-4	307	1E-4
	32	178	6E-5	–	–	–	–	178	6E-5	178	6E-5
	64	126	3E-6	–	–	–	–	124	2E-6	124	2E-6
SKY3D $tol = 10^{-8}$	2	610	4E-5	611	4E-5	611	4E-5	611	4E-5	638	1E-5
	4	420	2E-5	–	–	–	–	424	1E-5	418	2E-5
	8	228	1E-5	–	–	–	–	230	1E-5	228	2E-5
	16	134	1E-5	–	–	–	–	134	1E-5	134	1E-5
	32	87	1E-6	–	–	–	–	83	1E-5	83	1E-5
	64	53	6E-6	–	–	–	–	51	1E-5	51	1E-5
ANI3D $tol = 10^{-8}$	2	893	6e-5	893	6e-5	893	6e-5	893	6e-5	893	6e-5
	4	749	8e-5	749	8e-5	749	8e-5	749	8e-5	749	8e-5
	8	498	8e-5	506	9e-5	511	8e-5	498	7e-5	503	7e-5
	16	328	1e-4	–	–	–	–	326	1e-4	326	1e-4
	32	192	2e-4	–	–	–	–	192	1e-4	192	1e-4
	64	122	5e-5	–	–	–	–	122	4e-5	122	4e-5

Note that MSDO-CG did not converge when one of these A-orthonormalization combinations were used, CGS+CGS, CGS2+CGS2, CGS2+CGS, or CGS+CGS2 A-orthonormalization. The reason is that the seach directions are not A-orthogonal to satisfactory precision. And by Theorem 3.10, this implies that $r_k \not\perp \mathcal{K}_{k,t}$. Thus, nothing guarantees convergence since we have shown in section 3.1.3 that MSDO-CG converges faster than CG if $r_k \perp \mathcal{K}_{k,t}$. Moreover, we did not test combinations of MGS and QR factorizations since MGS is expensive in terms of communication compared to the other methods (section 5). But we tested MSDO-CG with MGS for comparison purposes. Note that when using MGS in Algorithm 1 we solve the $\alpha_k = (P_k^t W_k)^{-1}(P_k^t r_{k-1})$ and $\beta_k = (P_{k-1}^t W_{k-1})^{-1}(W_{k-1}^t r_{k-1})$ systems. Whereas when using CGS2+CholQR or CGS2+PreCholQR, we use $\alpha_k = (P_k^t r_{k-1})$ and $\beta_k = (W_{k-1}^t r_{k-1})$.

As shown in Table 2, MSDO-CG with MGS A-orthonormalization converges for all the tested matrices and as we increase t, the number of iterations needed for convergence decreases. As we mentioned earlier, MSDO-CG with CGS A-orthonormalization did not converge. Therefore, we replaced CGS+CGS with CGS+A-CholQR and with

CGS+Pre-CholQR A-orthonormalization. We notice that MSDO-CG with CGS+A-CholQR A-orthonormalization and MSDO-CG with CGS+Pre-CholQR A-orthonormalization have the same convergence behavior. For the matrices Poisson2D and Nh2D, both methods converge with the same number of iterations as MSDO-CG with MGS A-orthonormalization. However, for the matrix Sky2D, both methods did not converge. As for the matrices Sky3D and Ani3D, both methods converged only for $t = 2$ partitions, and $t = 2, 4, 8$ partitions, respectively. The reason for this difference in behavior for different matrices is the condition number ($\kappa = ||A||_2||A^{-1}||_2$). The condition number of the matrices Poisson2D and Nh2D is 6×10^3, whereas that of the matrices Sky3D, Ani3D, and Sky2D is 1×10^6 , 2×10^6, and 3×10^7, respectively. Although it was shown in [22] that Pre-CholQR A-orthonormalization is more stable than A-CholQR, however, MSDO-CG with CGS+A-CholQR A-orthonormalization and MSDO-CG with CGS+Pre-CholQR A-orthonormalization are both numerically unstable.

Thus, we replace CGS with CGS2, where the A-orthonormalization is performed twice for numerical stability. Then, the MSDO-CG with CGS2+A-CholQR A-orthonormalization and MSDO-CG with CGS2+Pre-CholQR A-orthonormalization converge as fast as MSDO-CG with MGS A-orthonormalization for all t and all the tested matrices. Hence, we conclude that CGS2+A-CholQR and CGS2+Pre-CholQR A-orthonormalizations are stable enough to be used in the MSDO-CG method (Algorithm 1). We exclude MGS A-orthonormaliztion since we have to solve two $t \times t$ systems at each iteration unlike when using CGS2+A-CholQR or CGS2+Pre-CholQR A-orthonormalization.

In Table 3, we compare the convergence behavior of the LRE-CG method with different orthonormalization schemes for orthonormalizing W against the $n \times tk$ matrix Q (MGS, CGS) and then orthonormalizing W against itself (MGS, CGS, TSQR (parallelizable tall and skinny QR)) and for different numbers of partitions $t = 2, 4, 8, 16, 32, 64$ that correspond to the maximum number of vectors added at each iteration to the enlarged Krylov subspace, $\mathcal{K}_{k,t}$. We start by testing the convergence of LRE-CG with MGS (MGS+MGS) orthonormalization. It converges for all the tested matrices since it is numerically stable, and the number of iterations needed for convergence decreases when increasing the number of partitions t. However, as mentioned in section 5, MGS is expensive in terms of communication when executed on t processors; it requires $O(tklog(t))$ messages for A-orthonormalizing t vectors against the previous tk vectors. Thus, we tested the LRE-CG method with CGS orthogonalization which requires sending $O(tlog(t))$ messages per iteration. The LRE-CG with CGS converges in the same number of iterations as LRE-CG with MGS for the matrices Poisson2D and Nh2D. However, for the other matrices, it does not converge for the given stopping criteria, except for $t = 2$ as shown in Table 3. The matrix $C = Q^t AQ$ is becoming close to singular, with $rank(C) < tk$, as the iterations proceed, and this is due to the loss of orthogonality in the CGS orthogonalization. The number of iterations in parentheses in Table 3 is not the number of iterations to convergence but it denotes the iteration at which the matrix C becomes close to singular.

In CA-GMRES [23], the authors use a TSQR factorization [5] for orthonormalizing the $n \times t$ tall and skinny matrix instead of CGS. They have shown that the combination of CGS for orthonormalizing W against Q and TSQR for orthonormalizing W is stable. We have tested LRE-CG with CGS and TSQR (CGS+TSQR) orthonormalization, and it has the same convergence behavior as LRE-CG with MGS (MGS+MGS) orthonormalization (Table 3). Thus, we conclude that MGS and CGS+TSQR orthonormalizations are stable enough to be used in the LRE-CG method [13].

TABLE 3

Comparison of the convergence of the LRE-CG method with different orthonormalization schemes, with respect to number of partitions t, with $x_0 = 0$. The number of iterations in parentheses is not the number of iterations for convergence but it denotes the iteration at which the $C = Q^t AQ$ matrix becomes close to singular.

		LRE-CG with different orthonormalization methods					
		MGS+MGS		CGS+CGS		CGS+TSQR	
	Pa	Iter	Err	Iter	Err	Iter	Err
POISSON2D $tol = 10^{-6}$	2	193	2E-5	193	2E-5	193	2E-5
	4	153	1E-5	153	1E-5	153	1E-5
	8	123	8E-6	123	8E-6	123	8E-6
	16	95	4E-6	95	4E-6	95	4E-6
	32	70	2E-6	70	2E-6	70	2E-6
	64	52	1E-6	52	1E-6	52	1E-6
NH2D $tol = 10^{-8}$	2	245	1E-7	245	1E-7	245	1E-7
	4	188	1E-7	188	1E-7	188	1E-7
	8	149	5E-8	149	5E-8	149	5E-8
	16	112	3E-8	112	3E-8	112	3E-8
	32	82	2E-8	82	2E-8	82	2E-8
	64	60	1E-8	60	1E-8	60	1E-8
SKY2D $tol = 10^{-8}$	2	1415	5E-04	1415	8E-4	1415	5E-04
	4	757	1E-4	(140)	–	754	1E-4
	8	398	1E-4	(112)	–	398	1E-4
	16	220	9E-5	(70)	–	220	1E-4
	32	126	5E-5	(51)	–	126	5E-5
	64	75	3E-5	(29)	–	75	4E-5
SKY3D $tol = 10^{-8}$	2	557	2E-5	570	1E-5	563	1E-5
	4	373	2E-5	(140)	–	377	1E-5
	8	211	1E-5	(54)	–	211	1E-5
	16	119	9E-6	(37)	–	119	9E-6
	32	69	9E-6	(18)	–	69	9E-6
	64	43	4E-6	(15)	–	42	1E-5
ANI3D $tol = 10^{-8}$	2	875	7e-5	875	7E-5	875	7e-5
	4	673	8e-5	(185)	–	673	8e-5
	8	449	1e-4	(116)	–	449	1e-4
	16	253	2e-4	(16)	–	253	2e-4
	32	148	2e-4	(9)	–	148	2e-4
	64	92	1e-4	(13)	–	92	1e-4

We did not test the SRE-CG versions with the different A-orthonormalization techniques. But one could use the CGS2+Pre-CholQR A-orthonormalization or the CGS2+A-CholQR A-orthonormalization, similarly to MSDO-CG. In Table 4, we compare the convergence of all the introduced enlarged Krylov subspace methods for the different t values with tolerance equal to 10^{-8}. We have tested the convergence of the SRE-CG versions with the CGS2+Pre-CholQR A-orthonormalization. And in the truncated SRE-CG2 version we A-orthonormalize W_k against the last k_{trunc} sets of t vectors, i.e., $W_{k-k_{trunc}}, \ldots, W_{k-2}, W_{k-1}$, where for testing purposes we set $k_{trunc} = 20$ and $k_{trunc} = 50$. But in practice the choice of k_{trunc} depends mainly on the available memory.

For the matrices POISSON2D and NH2D, all the SRE-CG versions have the same convergence rate. Thus SRE-CG is preferred due to its fixed memory requirements, similarly to CG. However, this is not the case for other matrices. It is clear that for the matrices SKY2D, SKY3D, and ANI3D, the larger k_{trunc} is, the better the convergence of the truncated SRE-CG2 method is. Moreover, truncated SRE-CG2 converges faster

TABLE 4
The convergence of the enlarged CG methods for tol $= 10^{-8}$.

	t	CG	SRE-CG	SRE-CG2 Trunc(20)	SRE-CG2 Trunc(50)	SRE-CG2	LRE-CG	MSDO-CG
POISSON2D $tol = 10^{-6}$	2	195	193	193	193	193	193	204
	4		153	153	153	153	153	167
	8		123	123	123	123	123	139
	16		95	95	95	95	95	121
	32		70	70	70	70	70	94
	64		52	52	52	52	52	69
NH2D $tol = 10^{-8}$	2	259	245	245	245	245	245	256
	4		188	188	188	188	188	208
	8		149	149	149	149	149	169
	16		112	112	112	112	112	138
	32		82	82	82	82	82	107
	64		60	60	60	60	60	77
SKY2D $tol = 10^{-8}$	2	5951	5595	5592	5477	1415	1415	1559
	4		4613	4529	4347	757	757	917
	8		2893	2730	2555	398	398	534
	16		1804	1640	1441	220	220	307
	32		995	863	678	126	126	178
	64		510	386	159	75	75	124
SKY3D $tol = 10^{-8}$	2	902	849	837	826	557	557	610
	4		745	741	691	373	373	420
	8		589	555	481	211	211	228
	16		434	394	323	119	119	134
	32		281	222	146	69	69	87
	64		157	101	42	43	43	53
ANI3D $tol = 10^{-8}$	2	4146	3876	3943	3899	875	875	893
	4		3565	3530	3477	673	673	749
	8		3175	2830	2700	449	449	498
	16		2303	1960	1705	253	253	328
	32		1653	1234	549	148	148	192
	64		930	483	248	92	92	122

than SRE-CG, and SRE-CG2 converges faster than truncated SRE-CG2 for all the t values. And all the SRE-CG versions converge faster than CG. What is interesting to note is that SRE-CG2 and LRE-CG have the same convergence rate for the matrices in our set. Thus the two methods are equivalent mathematically and numerically, although in LRE-CG we orthonormalize the basis and solve a system to obtain α_k, whereas in SRE-CG2 we A-orthonormalize the basis and compute a matrix-vector multiplication to get α_k. But both methods orthonormalize or A-orthonormalize the whole basis.

In Table 5, we compare the convergence behavior of MSDO-CG with MGS A-orthonormalization, SRE-CG2 with CGS2+Pre-CholQR A-orthonormalization, truncated SRE-CG2 with CGS2+Pre-CholQR A-orthonormalization and $k_{trunc} = 50$, coop-CG, and MSD-CG with respect to CG, for several matrices with different numbers of partitions $t = 2, 4, 8, 16, 32, 64$. The MSDO-CG, COOP-CG, SRE-CG2, and truncated SRE-CG2 have better convergence than CG, and SRE-CG2 has the best convergence. MSD-CG converges, but requires more iterations than CG, at least three times more iterations for the matrices SKY2D, SKY3D, ANI3D, and ELASTICITY3D. As for coop-CG, which starts with t different initial guesses and solves two systems of fixed size $t \times t$, its convergence is slightly better than MSDO-CG for the matrices POISSON2D, NH2D, and ELASTICITY3D. But it requires many more iterations than both MSDO-CG and SRE-CG for the other matrices (SKY2D, SKY3D, ANI3D).

TABLE 5
Comparison between the convergence of the different CG versions with respect to number of partions (t) or initial guesses for coop-CG with $x_0 = 0$.

		CG		coop-CG		MSD-CG		MSDO-CG		SRE-CG2		SRE-CG2 Trunc(50)	
	t	Iter	Err	Iter	Err	Iter	Err	Iter	Err	Iter	Err	Iter	Err
POISSON2D $tol = 10^{-6}$	2	195	2E-5	206	2E-7	235	3E-1	200	4E-5	193	2E-5	193	2E-5
	4			171	1E-7	252	7E-1	167	2E-5	153	1E-5	153	1E-5
	8			137	1E-7	245	7E-1	139	1E-5	123	8E-6	123	8E-6
	16			106	3E-8	249	7E-1	121	5E-6	95	4E-6	95	4E-6
	32			80	1E-8	240	7E-1	94	2E-6	70	2E-6	70	2E-6
	64			59	1E-8	253	7E-1	69	2E-6	52	1E-6	52	1E-6
NH2D $tol = 10^{-8}$	2	259	4E-7	206	2E-7	363	3E-1	256	1E-7	245	1E-7	245	1E-7
	4			179	1E-7	343	7E-1	208	1E-7	188	1E-7	188	1E-7
	8			157	2E-5	372	7E-1	169	8E-8	149	5E-8	149	5E-8
	16			107	2E-8	373	7E-1	138	6E-8	112	3E-8	112	3E-8
	32			81	2E-8	324	7E-1	107	2E-8	82	2E-8	82	2E-8
	64			59	1E-8	457	7E-1	77	1E-8	60	1E-8	60	1E-8
SKY2D $tol = 10^{-8}$	2	5951	4E-4	4893	2E-4	17907	3E-1	1559	8E-4	1415	5E-04	5477	2E-04
	4			3737	9E-5	66979	7E-1	917	4E-4	757	1E-4	4347	3E-05
	8			3391	1E-5	25298	7E-1	532	3E-4	398	1E-4	2555	2E-05
	16			2437	9E-6	23486	7E-1	307	1E-4	220	9E-5	1441	1E-05
	32			1406	4E-6	15448	7E-1	178	6E-5	126	5E-5	678	4E-06
	64			802	2E-6	23981	7E-1	126	3E-6	75	3E-5	159	1E-05
SKY3D $tol = 10^{-8}$	2	902	1E-5	795	8E-6	3070	2E-1	610	4E-5	557	2E-5	826	1E-05
	4			627	1E-5	11572	6E-1	420	2E-5	373	2E-5	691	8E-06
	8			542	4E-6	3207	7E-1	228	1E-5	211	1E-5	481	4E-06
	16			414	3E-6	4225	7E-1	134	1E-5	119	9E-6	323	1E-06
	32			290	1E-6	3149	7E-1	87	1E-6	69	9E-6	146	1E-06
	64			183	8E-7	2719	7E-1	53	6E-6	43	4E-6	42	1E-05
ANI3D $tol = 10^{-8}$	2	4187	4e-5	3584	5e-5	12404	2e-1	893	6e-5	875	7e-5	3899	4e-5
	4			3371	4e-5	17311	6e-1	749	8e-5	673	8e-5	3477	4e-5
	8			2865	4e-5	22339	7e-1	498	8e-5	449	1e-4	2700	5e-5
	16			2314	3e-5	21989	7e-1	328	1e-4	253	2e-4	1705	4e-5
	32			1615	2e-5	17042	7e-1	192	2e-4	148	2e-4	549	8e-5
	64			1002	1e-5	19257	1e-4	122	5e-5	92	1e-4	248	6e-5
ELASTICITY3D $tol = 10^{-8}$	2	1098	1e-7	744	1e-7	28708	6e-1	830	1e-7	652	1e-7	668	1e-7
	4			528	1e-7	18248	7e-1	621	1e-7	445	1e-7	457	1e-7
	8			417	1e-7	19603	7e-1	513	5e-8	321	8e-8	332	7e-8
	16			319	1e-6	11978	7e-1	388	5e-8	238	4e-8	248	5e-8
	32			268	5e-7	9594	7e-1	338	8e-5	168	5e-8	181	3e-8
	64			216	1e-6	7022	7e-1			116	1e-8	131	2e-8

Moreover, the results may vary depending on the t initial guesses that are used for the different matrices.

For the tested matrices, SRE-CG2 has slightly better convergence than MSDO-CG, since it uses the whole basis to define the new approximate solution rather than t search directions. For the matrices POISSON2D and NH2D, SRE-CG and MSDO-CG have almost the same convergence as CG for $t = 2$, and then as t is doubled the number of iterations needed for convergence decreases by 20% to 30%. For $t = 2$, SRE-CG2 requires 35% and 40% fewer iterations than CG for the matrices ELASTICITY3D and SKY3D, respectively. And as t is doubled, the number of iterations needed for convergence decreases by 25% to 30%, and 32% to 45%, respectively. For $t = 2$, SRE-CG2 requires 60% and 80% fewer iterations than CG for the matrices SKY2D and ANI3D, respectively. And as t is doubled, the number of iterations needed for convergence decreases by 45% to 50% and 25% to 40%, respectively.

As it is clear from the convergence tests, by doubling t, the number of iterations needed for convergence is not always reduced by 50% for all the matrices. However, as shown in the previous sections, the memory requirements for MSDO-CG, LRE-CG, and SRE-CG2, except for the matrix A, are $(tk + t + 2)n + 2t$, $(tk + 2)n + (tk)^2$, and $(tk + 2)n$, respectively, where n is the size of the matrix, and k is the number of computed iterations. As for the truncated SRE-CG2, SRE-CG, and CG, we only need to store $(tk_{trunc} + 2)n$, $(3t + 2)n$, $5n$ entries, respectively, where $2 < k_{trunc} < k \leq k_{\max}$. Thus, by doubling t, the memory requirements for MSDO-CG, LRE-CG, and SRE-CG2 for performing k iterations is at least doubled. But, when t is doubled, k decreases. Thus, the memory requirements increase and at most double, when t is doubled. Hence, t should be relatively small depending on the size of the matrix, on the performed iterations, and on the available memory. However, the memory requirements for truncated SRE-CG2 and SRE-CG are fixed, similarly to CG. Thus there is no memory restrictions on the value of t. In this case, t is chosen to obtain a numerically stable basis that leads to better convergence.

In this paper, we do not discuss preconditioning. But, similarly to the Krylov subspace methods, the main difference between the preconditioned and the unpreconditioned versions of MSDO-CG, LRE-CG, SRE-CG, and SRE-CG2 is that A is replaced by $\widehat{A} = L^{-1}AL^{-t}$ and b is replaced by $\widehat{b} = L^{-1}b$. Then, after finding the solution $\widehat{x}$ of the preconditioned system $\widehat{A}\widehat{x} = \widehat{b}$, the solution of the original system x is obtained by solving $L^t x = \widehat{x}$. Note that in MSDO-CG, SRE-CG, and SRE-CG2, the A-orthonormalization is replaced by $\widehat{A}$-orthonormalization, which is discussed in [24]. A detailed description of the preconditioned enlarged CG methods, specifically the preconditioned MSDO-CG and LRE-CG, can be found in the technical report [13]. Tables 6 and 7 in [13] compare the convergence behavior of the block Jacobi preconditioned MSDO-CG and LRE-CG to that of preconditioned CG (PCG). As for the preconditioned SRE-CG, SRE-CG2(20), SRE-CG2(50), and SRE-CG2, they converge in exactly the same number of iterations as the preconditioned LRE-CG. In summary, the preconditioned enlarged CG versions converge faster than PCG, but the difference in the number of iterations is fewer than that of the unpreconditioned versions. In addition, for an efficient preconditioner, the preconditioned SRE-CG seems to be the most promising enlarged CG version, since it converges in fewer iterations than PCG and has similar memory requirements.

5. Parallel model and expected performance. In this section, we present first the sequential timing of the kernels in the SRE-CG versions. Then, we briefly describe the parallelization of the MSDO-CG method and the SRE-CG methods with computed flops, number of messages and words sent, and the estimated parallel runtime. For a detailed discussion on the parallelization of MSDO-CG and LRE-CG refer to [13].

The estimated time for computing z flops is $\gamma_c z$, where γ_c is the inverse floating-point rate, also called the floating-point throughput (seconds per floating-point operation). The estimated time for sending a message of size k is $\alpha_c + \beta_c k$, where α_c is the latency (in seconds) and β_c is the inverse bandwidth (seconds per word). Hence, the estimated runtime of an algorithm with a total of z computed flops and s sent messages, each of size k, is the sum of their corresponding estimated times $\gamma_c z + \alpha_c s + \beta_c$.

The SRE-CG algorithms can be divided into four computational kernels or routines. The first routine is the matrix block of vector multiplications, $A * W_k$, which is computed after defining W_k at iteration $k \geq 1$. The second is the CGS2 A-

TABLE 6

Comparison between the runtime of SRE-CG methods with CGS2+A-CholQR A-orthonormalization with respect to number of partions (t). We show the total runtime in seconds for each of the routines $A * W_k$ (AW), CGS2 A-orthonormalization (CGS), A-CholQR A-orthonormalization (ACh), in addition to the total runtime (Tot) that includes the time for updating the variables.

	CG	SRE-CG					SRE-CG2 Trunc(20)				SRE-CG2 Trunc(50)			
		t	AW	CGS	ACh	Tot	AW	CGS	ACh	Tot	AW	CGS	ACh	Tot
NH2D	0.13	**2**	0.04	0.21	0.14	0.44	0.04	0.77	0.16	1.04	0.04	1.69	0.16	1.95
		8	0.08	0.62	0.30	1.05	0.07	2.19	0.28	2.59	0.08	4.39	0.28	4.80
		32	0.16	1.86	0.97	3.05	0.16	8.67	0.89	9.78	0.16	17.6	0.92	18.70
SKY3D	0.40	**2**	0.13	0.58	0.37	1.23	0.15	2.15	0.41	2.96	0.14	4.73	0.40	5.49
		8	0.34	1.94	1.22	3.68	0.28	7.04	0.79	8.28	0.23	12.50	0.67	13.55
		32	0.65	5.95	3.08	9.89	0.46	21.40	2.06	24.08	0.27	27.09	1.22	28.66
ANI3D	1.86	**2**	0.62	2.51	1.66	5.40	0.69	9.88	1.85	13.52	0.64	22.08	1.84	25.56
		8	1.84	10.30	5.02	18.03	1.54	38.57	4.35	45.49	1.49	82.53	4.20	89.23
		32	4.15	38.62	19.92	64.09	2.78	132.95	12.36	149.03	1.16	136.68	5.22	143.46

TABLE 7

Comparison between the runtime of SRE-CG methods with CGS2+Pre-CholQR A-orthonormalization with respect to number of partions (t). We show the total runtime in seconds for each of the routines $A * W_k$ (AW), CGS2 A-orthonormalization (CGS), Pre-CholQR A-orthonormalization (PCh), in addition to the total runtime (Tot) that includes the time for updating the variables.

	CG	SRE-CG					SRE-CG2 Trunc(20)				SRE-CG2 Trunc(50)			
		t	AW	CGS	PCh	Tot	AW	CGS	PCh	Tot	AW	CGS	PCh	Tot
NH2D	0.13	**2**	0.05	0.25	0.29	0.64	0.05	0.95	0.28	1.35	0.04	1.52	0.25	1.88
		8	0.08	0.61	0.55	1.29	0.08	2.19	0.54	2.85	0.08	4.38	0.53	5.03
		32	0.18	2.17	2.06	4.48	0.16	9.56	1.92	11.72	0.16	17.04	1.75	19.01
SKY3D	0.40	**2**	0.13	0.52	0.67	1.47	0.15	2.24	0.74	3.37	0.13	4.37	0.68	5.37
		8	0.32	1.78	1.68	3.94	0.25	6.36	1.51	8.30	0.23	12.24	1.35	13.96
		32	0.51	4.90	4.85	10.44	0.40	18.93	3.69	23.15	0.27	29.53	2.51	32.40
ANI3D	1.86	**2**	0.59	2.37	3.14	6.70	0.66	9.34	3.28	14.36	0.63	20.83	3.23	25.69
		8	1.94	10.35	9.73	23.09	1.33	34.07	7.65	43.94	1.32	74.91	7.72	84.84
		32	3.00	28.98	27.24	60.22	2.29	114.12	20.87	138.07	0.99	124.19	9.13	134.65

orthonormalization of W_k against Q, where in SRE-CG $Q = [W_{k-2}, W_{k-1}]$, in SRE-CG2 $Q = [W_1, W_2, \ldots, W_{k-2}, W_{k-1}]$, and in truncated SRE-CG2 $Q = [W_{k-k_{trunc}}, W_{k-k_{trunc}+1}, \ldots, W_{k-2}, W_{k-1}]$ for a fixed k_{trunc} that satisfies $3 \leq k_{trunc} < k \leq k_{\max}$. The third routine is the A-orthonormalization of W_k using either A-CholQR or Pre-CholQR. And the fourth routine is updating the variables α_k, x_k, r_k, and ρ_k. Note that the time required for forming the matrices Q as described above, and W_k as $W_k = AW_{k-1}$ for $k > 1$ and $W_1 = \mathcal{T}(r_0)$, is not reported.

In Table 6, we show the total sequential time in seconds for solving the systems using SRE-CG, SRE-CG2(20), and SRE-CG2(50) methods with CGS2+A-CholQR A-orthonormalization. In Table 7, we show the total sequential time in seconds for solving the systems using SRE-CG, SRE-CG2(20), and SRE-CG2(50) methods with CGS2+PreCholQR A-orthonormalization. In both tables we show the total sequential time needed for convergence for $t = 2, 8, 32$, and the total time needed for each of the aforementioned routines ($A * W_k$, CGS2, A-CholQR/PreCholQR), except for update, which is included in the total sequential time. For the matrix NH2D, update's time is almost constant and takes less than 0.08 seconds. For the matrices SKY3D and ANI3D, update's time decreases as t increases, except in the case of SRE-CG. The

most time consuming part in MATLAB is the A-orthonormalization, specifically the CGS2 A-orthonormalization. Thus, as t increases, and as k_{trunc} increases, the A-orthonormalization's sequential time increases, and so does the total runtime. Thus, it is normal in MATLAB that the sequential SRE-CG methods require much more time to converge as compared to the sequential CG. However, it is expected that in parallel, the SRE-CG methods will require much less time to converge, as discussed at the end of this section.

As shown in Table 4, the different SRE-CG versions with CGS2+Pre-CholQR A-orthonormalization converge in the same number of iterations for the system NH2D. Moreover, using CGS2+A-CholQR A-orthonormalization does not affect the convergence of the SRE-CG versions for the system NH2D. Thus it can be used as a reference case. By comparing the MATLAB timing of the A-CholQR and Pre-CholQR in Tables 6 and 7, it is clear that Pre-CholQR requires around double the time of A-CholQR. On the other hand, in SRE-CG2(20) and SRE-CG2(50), the total flops performed in CGS2 A-orthonormalization, is $\frac{k_{trunc}(2k-k_{trunc}+1)}{2(2k-1)}$ times those performed in SRE-CG, assuming that all three methods converge in k iterations with $k_{trunc} = 20$ or 50. For example, in SRE-CG2(20), the flops performed in CGS2 are around 9 times that of SRE-CG, whereas, in SRE-CG2(50), the flops performed in CGS2 range between 18 and 22 times that of SRE-CG, depending on the t value. However, the total sequential time needed for the CGS2 A-orthonormalization throughout the SRE-CG2(20) and SRE-CG2(50) iterations is at most 4.5 and 9 times that of SRE-CG, respectively, as shown in Tables 6 and 7. This is due to the communication reduction by performing more operations per memory access.

However, this is not the case for the other matrices, SKY3D, and ANI3D. First, SRE-CG2(50) converges faster than SRE-CG2(20) which converges faster than SRE-CG, as shown in Table 4. Moreover, in some cases, using CGS2+Pre-CholQR A-orthonormalization produces a numerically more stable basis than when using CGS2+A-CholQR A-orthonormalization. This implies a faster convergence in terms of iterations. However, performing one Pre-CholQR factorization is more expensive in terms of flops than performing an A-CholQR factorization. This is clear in Table 7, where CGS2 A-orthonormalization requires less time than that in Table 7 for most t values, but PreCholQR requires more time than A-CholQR. For the SKY2D matrix, the SRE-CG methods with CGS2+Pre-CholQR have a similar runtime to those with CGS2+A-CholQR A-orthonormalization except for SRE-CG2(50) with $t = 32$. This is not the case for the ANI3D matrix, where the SRE-CG methods with CGS2+Pre-CholQR converge in less time than the corresponding SRE-CG methods with CGS2+A-CholQR, for $t = 32$. Note that for $t = 32$, the SRE-CG2(50) method converges in less time than the SRE-CG2(20).

For simplicity, we assume that the algorithms are executed on a distributed memory machine formed by t processors, where t corresponds to the number of vectors computed at each iteration. We partition the graph of A into t subdomains using k-way partitioning or another graph partitioning. We denote by δ_i for $i = 1, 2, \ldots, t$, the subsets of indices obtained from the partitioning. That is $\delta_i \cap \delta_j = \phi$ for all $i \neq j$, $\cup_{j=1}^{t} \delta_j = \{1, 2, 3, \ldots, n\}$, and $|\delta_i| \approx \frac{n}{t}$. Then each processor i is assigned the $\frac{n}{t} \times n$ rowwise part of the matrix A ($A(\delta_i, :) = A(:, \delta_i)$ since A is SPD), the $\frac{n}{t} \times 1$ rowwise part of the vector b ($b(\delta_i)$), and the vector $x_0(\overline{\delta}_i)$, where $\overline{\delta}_i = Adj(G(A), \delta_i)$ is the adjacent of δ_i in the graph of A. Processor i computes $x_k(\delta_i)$.

However, for performance reasons and due to the multicore nature of most architectures, it is possible to use a number of processors greater than t, preferably a multiple of t. In this case, we start by partitioning the graph of A into t subdomains

using k-way partitioning or another graph partitioning, where δ_i for $i = 1, 2, \ldots, t$ are the subsets of indices obtained from the partitioning. This partitioning is used to define the $T(.)$ operator and eventually the enlarged Krylov subspace. Assuming that we have jt processors, then every j processors are assigned an $\frac{n}{t} \times n$ rowwise part of the matrix A, $A(\delta_i, :)$, $\frac{n}{t} \times 1$ rowwise part of the vector b ($b(\delta_i)$) and the vector $x_0(\overline{\delta}_i)$, and should output $x_k(\delta_i)$. In other words, we partition each of our t subdomains into j nonoverlapping subdomains to obtain a total of jt subdomains with set of indices $\delta_{i,l}$, where $i = 1, 2, \ldots, t$, $l = 1, 2, \ldots, j$, and $\delta_i = \cup_{l=1}^{j} \delta_{i,l}$. Then, in the discussion below on the number of messages and words sent, $log(t)$ is replaced by $log(jt)$, and $\frac{n}{t}$ is replaced by $\frac{n}{jt}$.

In MSDO-CG, SRE-CG, SRE-CG2, and truncated SRE-CG2, we A-orthonormalize the basis. As mentioned in section 4, MGS, CGS2+A-CholQR, and CGS2+Pre-CholQR A-orthonormalizations are numerically the most stable and allow the convergence of MSDO-CG for the matrices in our test set. As discussed in Appendix B of the technical report [13], the most parallelizable versions of MGS, Algorithms 14 and 15, require sending $(tk + 1)log(t)$ and $2(t - 1)log(t)$ messages, respectively, whereas CGS2, Algorithm 22 in [13], requires sending $4log(t)$ messages. On the other hand, Algorithm 25 from [13] of A-CholQR requires sending $log(t)$ messages, and Pre-CholQR Algorithm 27 requires sending $3log(t)$ messages. The CGS2+A-CholQR and CGS2+Pre-CholQR A-orthonormalizations can be called communication avoiding since they require sending $5log(t)$ and $7log(t)$ messages, respectively, unlike the MGS A-orthonormalization. Since both methods are stable and CGS2+A-CholQR requires less communication, it can be used in the four mentioned CG versions.

In Algorithms 1, 2, and 3, we have two types of communication. The first is an "all reduce" communication that requires synchronization between all the processors and is equivalent to $log(t)$ messages, each of the same size (refer to [30]). For example, the dot products require "all reduce" communication. The second type of communication is a point-to-point communication between each processor i and its m_i neighboring processors for computing a matrix block of vectors muliplication, specifically $A\mathcal{T}(r)$. Moreover, several communication steps could be overlapped with other computations. For a detailed description refer to [13].

The estimated time of k iterations of Algorithm 1 in parallel with t processors is

$$\begin{aligned}\text{Time}_{\text{MSDO-CG}}(k) \approx \gamma_c(2\tfrac{\text{nnz}}{t} + 12ntk + 10nt + 17n)k + \alpha_c(7log(t) + m_{MB})k \\ + \beta_c(\tfrac{n}{t}m_{MB} + t^2klog(t))k,\end{aligned}$$

where nnz is the number of nonzero entries in A, and $m_{MB} = max\{m_i| \quad i = 1, 2, \ldots, t\}$, the largest number of neighboring processors, with $m_i \leq m_{MB} \leq (t - 1)$ for all i.

The estimated time of k iterations of Algorithm 2 in parallel with t processors is

$$\begin{aligned}\text{Time}_{\text{SRE-CG}}(k) \approx \gamma_c(2\text{nnz} + 24nt + 5n)k + \alpha_c(6log(t) + m_{MB})k \\ + \beta_c(\tfrac{n}{t}m_{MB} + 4t^2log(t))k.\end{aligned}$$

And that of k iterations of Algorithm 3 in parallel with t processors is

$$\begin{aligned}\text{Time}_{\text{SRE-CG2}}(k) \approx \gamma_c(2\text{nnz} + 12ntk + 5n)k + \alpha_c(6log(t) + m_{MB})k \\ + \beta_c(\tfrac{n}{t}m_{MB} + t^2klog(t))k.\end{aligned}$$

The SRE-CG and SRE-CG2 methods exchange fewer messages than the MSDO-CG method. Moreover, the SRE-CG method sends fewer words and computes fewer

flops than the SRE-CG2 method. Hence, it is clear that computing k iterations of the SRE-CG method requires less time than MSDO-CG and SRE-CG2. However, as portrayed in the convergence section, for some matrices SRE-CG requires many more iterations than SRE-CG2 and MSDO-CG to converge. Hence it is not possible to claim that one of these methods will always be faster than the others in practice. For example, for the matrices Poisson2D and Nh2D, SRE-CG is the method to be used. But for the matrice Sky2D and Ani3D, SRE-CG2 or the truncated SRE-CG2 might be faster than SRE-CG.

6. Conclusion. In this paper we have introduced several new iterative methods, MSDO-CG, LRE-CG, SRE-CG, SRE-CG2, and truncated SRE-CG2, which are based on the enlarged Krylov subspace. After introducing the related existing methods (B-CG, coop-Cg, and MCD-CG), we have defined the properties of the enlarged Krylov subspace, derived the new methods in the context of projection CG versions, provided parallel versions that reduce communication, and shown that the methods converge at least as fast as classical CG in exact precision arithmetic. The convergence results show that they also converge faster than CG in finite precision arithmetic.

MSDO-CG is a variation of the MSD-CG version, where we A-orthonormalize the t search directions against previous directions and against each others. Due to the A-orthonormalization, we lose the short recurrence property of CG and we are obliged to save all the tk_c search directions, where k_c is the number of iterations till convergence. In LRE-CG we start by building an orthonormal basis for the enlarged Krylov subspace, then we use the whole basis to update the solution. The main difference between both methods in terms of performance, is that at each iteration of MSDO-CG, we use t search directions to update the new approximate solution, whereas in LRE-CG, at each iteration i, we use the entire basis formed by ti vectors to update the approximate solution and we solve a $ti \times ti$ system. However, this use of the whole basis leads to a relatively faster convergence than MSDO-CG. One way to limit this increasing cost is by restarting LRE-CG after some iterations.

Another alternative is to A-orthonormalize the basis rather than orthonormalizing it. In this case, we get three short recurrence enlarged CG methods, where the approximate solution is updated using the last t basis vectors. The difference between the three methods, SRE-CG, SRE-CG2, and truncated SRE-CG2, is in the A-orthonormalization of the basis. In the SRE-CG method, the t newly computed basis vectors at iteration i, are only A-orthonormalized against the previous $2t$ vectors. This limits the memory needed but affects the convergence of SRE-CG which, for some matrices, requires more iterations than MSDO-CG and LRE-CG to converge. In the SRE-CG2 method, the t newly computed basis vectors at iteration i, are A-orthonormalized against all the previous basis vectors. This leads to an identical convergence behavior as LRE-CG. In the truncated SRE-CG2 method, the t newly computed basis vectors at iteration i, are A-orthonormalized against a subset of the previous basis vectors, defined based on the available memory. This version converges faster than SRE-CG for most matrices, and requires less memory than SRE-CG2.

Although each iteration of the MSDO-CG, LRE-CG, SRE-CG, SRE-CG2, and truncated SRE-CG2 methods is at least t times more expensive than the CG iteration in terms of flops, as shown in section 5, these methods use less communication, and Blas2 and Blas3 operations that can be parallelized in a more efficent way than the dot products in CG. Moreover, the introduced methods converge faster than CG in terms of iterations as shown in section 4.

Our future work will focus on implementing, testing, and comparing the runtime of the introduced enlarged CG versions on parallel machines. We will also test these methods on other real applications' matrices, and with different preconditioners. We will also derive and test other enlarged Krylov methods, like enlarged GMRES which has been derived but not tested yet.

REFERENCES

[1] A. Bhaya, P. Bliman, G. Niedu, and F. A. Pazos, *A cooperative conjugate gradient method for linear systems permitting multithread implementation of low complexity*, in Proceedings of the 51th IEEE Conference on Decision and Control, CDC 2012, Maui, HI, IEEE, Piscataway, 2012, pp. 638–643.

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

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

[4] A. T. Chronopoulos and W. Gear, *s-step iterative methods for symmetric linear systems*, J. Comput. Appl. Math., 25 (1989), pp. 153–168.

[5] J. Demmel, L. Grigori, M. Hoemmen, and J. Langou, *Communication-optimal parallel and sequential QR and LU factorizations*, SIAM J. Sci. Comput., 34 (2012), pp. A206–A239.

[6] J. Demmel, M. Heath, and H. van der Vorst, *Parallel numerical linear algebra*, Acta Numer., 2 (1993), pp. 111–197.

[7] J. Erhel, *A parallel GMRES version for general sparse matrices*, Electron. Trans. Numer. Anal., 3 (1995), pp. 160–176.

[8] R. Fletcher, *Conjugate gradient methods for indefinite systems*, in Numerical Analysis, G.A. Watson, ed., Lecture Notes in Math. 506, Springer, Berlin, 1976, pp. 73–89.

[9] P. Ghysels, T. J. Ashby, K. Meerbergen, and W. Vanroose, *Hiding global communication latency in the GMRES algorithm on massively parallel machines*, SIAM J. Sci. Comput., 35 (2013), pp. C48–C71.

[10] G. H. Golub and D. P. O'Leary, *Some history of the conjugate gradient and Lanczos algorithms:* 1948–1976, SIAM Rev., 31 (1989), pp. 50–102.

[11] P. Gosselet, D. Rixen, F. Roux, and N. Spillane, *Simultaneous-FETI and Related Block Strategies: Robust Domain Decomposition Methods for Engineering Problems*, preprint, hal-01056928V1, 2014.

[12] L. Grigori and S. Moufawad, *Communication Avoiding ILU*0 *Preconditioner*, SIAM J. Sci. Comput., 37 (2015), pp. C217–C246.

[13] L. Grigori, S. Moufawad, and F. Nataf, *Enlarged Krylov Subspace Conjugate Gradient Methods for Reducing Communication*, preprint, hal-01065985, 2014.

[14] L. Grigori, F. Nataf, and S. Yousef, *Robust Algebraic Schur Complement Preconditioners Based on Low Rank Corrections*, preprint, hal-01017448, 2014.

[15] W. Gropp, *Update on Libraries for Blue Waters*, http://jointlab.pc.ncsa.illinois.edu/events/workshop3/pdf/presentations/Gropp-Update-on-Libraries.pdf.

[16] T. Gu, X. Liu, Z. Mo, and X. Chi, *Multiple search direction conjugate gradient method* I: *Methods and their propositions*. Int. J. Comput. Math., 81 (2004), pp. 1133–1143.

[17] F. Hecht, *New development in freefem++*, J. Numer. Math., 20 (2012), pp. 251–265.

[18] M. R. Hestenes and E. Stiefel, *Methods of conjugate gradients for solving linear systems*, J. Res. Natl. Bur. Stand., 49 (1952), pp. 409–436.

[19] M. Hoemmen, *Communication-Avoiding Krylov Subspace Methods*, Ph.D. thesis, EECS Department, University of California, Berkeley, CA, 2010.

[20] G. Karypis and V. Kumar, *Metis* 4.0: *Unstructured Graph Partitioning and Sparse Matrix Ordering System*, Technical report, Department of Computer Science, University of Minnesota, Minneapolis, MN, 1998.

[21] C. Lanczos, *Solution of systems of linear equations by minimized iterations*, J. Res. Natl. Bur. Stand, 49 (1952), pp. 33–53.

[22] B. Lowery and J. Langou, *Stability Analysis of QR factorization in an Oblique Inner Product*. preprint, ArXiv:1401-5171, 2014.

[23] M. Mohiyuddin, M. Hoemmen, J. Demmel, and K. Yelick, *Minimizing communication in sparse matrix solvers*, in Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis, SC 09, ACM, New York, 2009, pp. 1–12.

[24] S. Moufawad, *Enlarged Krylov Subspace Methods and Preconditioners for Reducing Communication*, Ph.D. thesis, Université Pierre et Marie Curie, Paris, 2014.

[25] D. P. O'Leary, *The block conjugate gradient algorithm and related methods*, Linear Algebra Appl., 29 (1980), pp. 293–322.
[26] D. J. Rixen, *Substructuring and Dual Methods in Structural Analysis*, Ph.D. thesis, Université de Liège, Liège, Belgium, 1997.
[27] M. Rozloznik, M. Tuma, A. Smoktunowicz, and J. Kopal, *Numerical stability of orthogonalization with a non-standard inner product*, BIT, 52 (2012), pp. 1035–1058.
[28] Y. Saad, *Analysis of augmented Krylov subspace methods*, SIAM J. Matrix Anal. Appl., 18 (1997), pp. 435–449.
[29] Y. Saad and M. H. Schultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Stat. Comput., 7 (1986), pp. 856–869.
[30] R. Thakur and W. Gropp, *Improving the performance of Collective Operations in MPICH*, in Proceedings of the 10th European PVM/MPI Users, 2003, Springer, New York, pp. 257–267.
[31] H. A. Van der Vorst, *Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems*, SIAM J. Sci. Stat. Comput., 13 (1992), pp. 631–644.
[32] J. Van Rosendale, *Minimizing inner product data dependencies in conjugate gradient iteration*, Techical report 172178, ICASE-NASA, 1983.
[33] H. F. Walker, *Implementation of the GMRES method using Householder transformations*, SIAM J. Sci. Stat. Comput., 9 (1998), pp. 152–163.