NLAFET
Parallel Numerical Linear Algebra for Future Extreme Scale Systems

H2020–FETHPC–2014: GA 671633

D4.2

Analysis and algorithm design

April 2017

DOCUMENT INFORMATION

Scheduled delivery	2017-04-30
Actual delivery	2017-04-28
Version	1.0
Responsible partner	INRIA

DISSEMINATION LEVEL

PU — Public

REVISION HISTORY

Date	Editor	Status	Ver.	Changes
2016-04-01	Inria members	Draft	0.1	Initial version of document produced
2016-04-24	Inria members	Final	1	Final revised version of document produced

AUTHOR(S)

Simplice Donfack, Laura Grigori, Olivier Tissot, INRIA

INTERNAL REVIEWERS

Carl Christian Kjelgaard Mikkelsen, UMU
Stocje Nakov, STFC

COPYRIGHT

ACKNOWLEDGEMENTS

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under the grant agreement number 671633.

Table of Contents

List of Figures

List of Tables

1 Executive summary

In this deliverable, we introduce some new algorithms for solving a symmetric positive definite linear system. On massively parallel machines, communication is the performance bottleneck. That is why those algorithms are designed in order to reduce the communication in parallel.

On one hand, we explain a method to reduce the number of search directions adaptively during the iterations of Enlarged Conjugate Gradient (ECG) [1, 2]. This method is cheap because its arithmetic overhead is low, just a SVD decomposition on a small matrix, and it does not need any communication. It is also completely algebraic and could be applied to any block Krylov Conjugate Gradient variant. On the other hand, we detail the construction and application of LORASC preconditioner [3], a robust and algebraic preconditioner. We show how it can be built and applied in parallel. To assess the effectivity of these methods, we perform several numerical experiments on a set of matrices including some coming from EDF CFD code *Code_Saturne* which is one of the industrial application of NLAFET (see *Deliverable D5.1: Requirements Analysis*). Numerical results on a set of matrices arising from the discretization by the finite element method of linear elasticity models illustrate the robustness, and show the fast convergence and efficiency of LORASC preconditioner. When testing ECG with or without adaptive reduction of the search directions, the numerical results show that our method allows to reduce the computational cost of ECG while maintaining a convergence rate very close to the one of ECG, and much better than the one of PCG.

To summarize, we introduce two methods that give promising results for solving SPD linear systems in parallel. These methods are currently implemented in parallel and they will be tested on massively parallel machines. Those results and the associated source code will be presented in details in *Deliverable D4.3*. In particular, we will test the implementation on *Code_Saturne* test cases and for the Cosmic Microwave Background problem in astrophysics (see *Deliverable 5.1: Requirements Analysis*). In addition, we will consider using LORASC as a preconditioner for ECG.

2 Introduction

The *Description of Action* document states for Deliverable 4.2:

> "*Analysis and algorithm design*
> Report on novel Krylov methods and multilevel preconditioners, focusing on numerical efficiency and theoretical properties."

This deliverable is in the context of Task 4.2 (Iterative methods) and Task 4.3 (Preconditioners).

In this deliverable, we discuss communication avoiding Krylov subspace iterative methods for solving large sparse linear systems of equations $Ax = b$, where A is a symmetric positive definite (SPD) matrix. We consider Krylov subspace methods based on Conjugate Gradient (CG) [7]. The convergence rate of CG depends on the condition number of the matrix A and on the distribution of its eigenvalues [15]. The convergence rate of Krylov subspace methods is often accelerated by seeking a preconditioner M such that the preconditioned matrix $M^{-1}A$ has better spectral properties and the linear system $M^{-1}Ax = M^{-1}b$ converges faster. Our research focuses on two mains aspects: design reformulations of Krylov-based iterative methods that, at the cost of more computation, will enable a reduction in the number of global communications with respect to classic formulations, and design communication-avoiding preconditioners that are efficient in accelerating the iterative method but also reduce communication. Our contributions can be summarized as follows.

First, we recall a novel approach introduced by *Grigori, Moufawad and Nataf* [1] for reducing the communications 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 solution of the linear system is sought in the enlarged subspace, which is a superset of the classic Krylov subspace. The construction of this enlarged space involves BLAS3 operations instead of BLAS1 and BLAS2 operations thus it increases arithmetic intensity and does not cost much more than constructing the corresponding classic Krylov subspace. Furthermore, as the solution is searched in a larger space, the number of iterations for finding an approximate solution is lower than with CG. Hence, the number of global synchronizations is reduced. Then, we explain a method, introduced in [2], for reducing adaptively the number of search directions during the iterations of Enlarged Conjugate Gradient.

Second, we introduce a robust algebraic preconditioner that can be efficiently built and applied in parallel. The graph of the matrix is partitioned by using k-way partitioning with vertex separators into N disjoint domains and a separator formed by the vertices connecting the N domains. The obtained permuted matrix has a block arrow structure. The preconditioner relies on the Cholesky factorization of the first N diagonal blocks and on approximating the Schur complement corresponding to the separator block. The approximation of the Schur complement involves the factorization of the last diagonal block and a low rank correction obtained by solving a generalized eigenvalue problem. The preconditioner can be build and applied in parallel. Our numerical results illustrate the robustness and the efficiency of our approaches.

The rest of the document is organized as follows. In section 3 we present enlarged Krylov methods. In section 4, we consider the LORASC preconditioner, discuss its general properties and its application. In Section 5, we present the results on a set of matrices arising from the discretization by the finite element methods of linear elasticity models, and illustrate the robustness and efficiency of our approaches. Finally, in section 6 we give a conclusion, and present future work.

3 Enlarged Conjugate Gradient

Block Krylov methods were introduced for the Conjugate Gradient in 1980 by *O'Leary* with the Block Conjugate Gradient method [12] (Block CG). She was motivated by solving linear systems with several right hand sides. She has shown theoretically that this method can converge significantly faster than the classical Conjugate Gradient.

3.1 Derivation

Following *Gutknecht at al.* [5] Block Krylov subspaces are defined as,

$$\mathscr{K}_k^{\square}(A,R_0) := \mathrm{span}^{\square}\left\{R_0, AR_0, \ldots, A^{k-1}R_0\right\} \tag{3.1}$$

$$:= \left\{\sum_{s=0}^{k-1} A^s R_0 \gamma_s \text{ such that } \forall s \in \{0,\ldots,k-1\},\ \gamma_s \in \mathbb{R}^{t\times t}\right\}. \tag{3.2}$$

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

$$X_k \in \mathscr{K}_k^{\square} + X_0, \tag{3.3}$$

$$R_k \perp \mathscr{L}_k^{\square}, \tag{3.4}$$

where $\mathscr{L}_k^{\square}$ is a subspace which has the same size as $\mathscr{K}_k^{\square}$. The first equation (3.3) is called the subspace condition and the second one (3.4) is called the Petrov-Galerkin condition.

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

$$\phi(X_k) = \min_{X \in \mathscr{K}_k^{\square}} \phi(X), \tag{3.5}$$

where

$$\phi(X) = \frac{1}{2} X^\top A X - B^\top X, \tag{3.6}$$

$$\nabla\phi(X) = AX - B. \tag{3.7}$$

As in gradient methods, the new solution at iteration k is defined as $X_k = X_{k-1} + P_k\alpha_k$, where P_k represent the descent directions and α_k is the step. One important property of the Block Conjugate Gradient is the A-orthonormality (or conjugacy) of the descent directions, that is $P_i^\top A P_j = 0$ when $i \neq j$.

As shown in [2] there exists two variants for constructing the search directions, Orthomin (see [2]) and Orthodir (Algorithm 1). In practice, Orthodir is more stable even if it is more expensive than Orthomin. That is why we focus on Orthodir in this report.

Algorithm 1 Block CG: orthodir

```
Require: $A$, $B$, $X_0$, $k_{\max}$, $\varepsilon_{\text{solver}}$
Ensure: $||B - AX|| < \varepsilon_{\text{solver}}$ or $k = k_{\max}$
 1: $R_0 = B - AX_0$
 2: $P_0 = 0$
 3: $P_1 =$ A-orthonormalize($R_0$)
 4: $k = 1$
 5: while $||R_{k-1}|| > \varepsilon_{\text{solver}}||B||$ and $k < k_{\max}$ do
 6:     $\alpha_k = P_k^\top R_{k-1}$                                          ▷ $t \times t$ matrix
 7:     $X_k = X_{k-1} + P_k\alpha_k$                                       ▷ $n \times t$ matrix
 8:     $R_k = R_{k-1} - AP_k\alpha_k$                                      ▷ $n \times t$ matrix
 9:     $P_{k+1} = AP_k - P_kP_k^\top AAP_k - P_{k-1}P_{k-1}^\top AAP_k$      ▷ $n \times t$ matrix
10:     $P_{k+1} =$ A-orthonormalize($P_{k+1}$)
11:     $k = k + 1$
12: end while
```

Even if the Block CG method was initially used to solve linear systems with several right hand sides, it is possible to use a block method to solve a linear system with only one right hand side. In the following we present different ways to use Block CG in order to solve a system with only one right hand side.

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

We denote 1_t a row of size $1 \times t$ and full of ones. We refer to the method defined in [11] by *Nikishin and Yeremin* as *BCG*. In this method, X_0 and B are choosen as $X_0 = x_0 1_t$ and $B^{(i)} \sim \mathscr{U}(0,1)$, $\forall t \geq i > 1$ and $B^{(1)} = b$. In that case, the method is stopped as soon as the first column of R_k has a norm smaller than $\varepsilon_{\text{solver}}||b||$. The solution is given by the first column of X. In [1] *Grigori, Moufawad and Nataf* use a domain decomposition approach to define enlarged Krylov subspaces and the associated *ECG* method. Given a splitting decomposition represented by the operator T,

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

where x is a vector of size n and $T(x)$ is of size $n \times t$. The first columns of $T(x)$ contains the first components of x and so on for the following columns of $T(x)$. The initial residual denoted r_0, the corresponding enlarged Krylov subspace is defined as

$$\mathscr{K}_{k,t} = \text{span}^{\square}\{T(r_0), AT(r_0), \ldots, A^{k-1}T(r_0)\}. \tag{3.8}$$

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

3.2 Adaptive reduction of the search directions

In this section, we present an approach for reducing the block size in the Orthodir method during the iterations defined in [2]. This is a technique known as deflation in block Krylov methods [5, 11, 13].

Indeed, as explained in the survey [5] the key idea to reduce the block size is to monitor the rank of R_{k-1}. Once R_{k-1} becomes rank deficient, it means that there exists a vector v of dimensions $t \times 1$ such that,

$$R_{k-1}v = 0, \tag{3.9}$$

and,

$$R_k v = R_{k-1}v - AP_k\alpha_k v \tag{3.10}$$
$$= 0 + AP_kP_k^\top R_{k-1}v \tag{3.11}$$
$$= 0. \tag{3.12}$$

It follows that $R_iv = 0$ for $i \geq k-1$. In other words, $X_{k-1}v$ has already converged at iteration $k-1$ because $X_{k-1}v = A^{-1}Bv$. For $i \geq k-1$, there exists a linear combination (independent of i) of columns of X_i denoted X_iv such that X_iv remains constant. As a consequence, P_kv is not useful to compute the approximate solution and the idea is to remove this search direction in the next iterations. As R_{k-1} is an $n\times t$ matrix with n large, it is preferable to avoid computing the rank of R_{k-1} directly. Our approach is based on computing the rank of $\alpha_k = P_k^\top R_{k-1}$. This is similar to the idea developed by *Robbé and Sadkane* in [13].

In practice, the case, where α_k becomes exactly rank deficient (also denoted *exact breakdown* or *lucky breakdown*), is very rare and it is preferable to detect when α_k becomes nearly rank deficient (also denoted *inexact breakdown*) [5, 13, 11].

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

$$\alpha_k = U_k\Sigma_kV_k^\top, \tag{3.13}$$

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

$$U_k\Sigma_kV_k^\top = \begin{pmatrix} U_k^{(1)} & U_k^{(2)} \end{pmatrix} \begin{pmatrix} \Sigma_k^{(1)} & 0 \\ 0 & \Sigma_k^{(2)} \end{pmatrix} \begin{pmatrix} V_k^{(1)^\top} \\ V_k^{(2)^\top} \end{pmatrix}, \tag{3.14}$$

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

Now let us define,

$$P_k^{(1)} = P_kU_k^{(1)}, \tag{3.15}$$
$$P_k^{(2)} = P_kU_k^{(2)}. \tag{3.16}$$

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

$$P_{k+1}U_k^{(1)} = AP_k^{(1)} - \begin{pmatrix} P_k^{(1)} & P_k^{(2)} \end{pmatrix} \begin{pmatrix} P_k^{(1)^\top} \\ P_k^{(2)^\top} \end{pmatrix} AAP_k^{(1)} - P_{k-1}P_{k-1}^\top AAP_k^{(1)} \tag{3.17}$$
$$= AP_k^{(1)} - P_k^{(1)}P_k^{(1)^\top}AAP_k^{(1)} - P_k^{(2)}P_k^{(2)^\top}AAP_k^{(1)} - P_{k-1}P_{k-1}^\top AAP_k^{(1)} \tag{3.18}$$

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

This allows us to define the new search directions the first time the size of the block is reduced. It is possible to generalize this idea when the size is reduced several times (possibly until it is equal to one) leading to Algorithm 2 [2]. It is also possible to use this idea to derive an adaptive variant of Orthomin [2].

Algorithm 2 Orthodir with block size reduction

```
Require: $A$, $B$, $X_0$, $k_{\max}$, $\varepsilon_{\text{solver}}$, $\varepsilon_{\text{def}}$
Ensure: $||B - AX|| < \varepsilon_{\text{solver}}$ or $k = k_{\max}$
 1: $R_0 = B - AX_0$
 2: $P_0 = 0$
 3: $P_1 =$ A-orthonormalize($R_0$)
 4: $k = 1$
 5: $H = 0$
 6: while $||R_{k-1}|| < \varepsilon_{\text{solver}}||B||$ and $k < k_{\max}$ do
 7:     $\alpha_k = P_k^\top R_{k-1}$
 8:     $[U_k, \Sigma_k, V_k] = \text{svd}(\alpha_k)$
 9:     $s_k =$ number of singular values of $\alpha_k$ bigger than $\varepsilon_{\text{def}}$
10:     if $s_k < s_{k-1}$ then
11:         $U_k^{(1)} = U_k(:, 1:s_k)$
12:         $U_k^{(2)} = U_k(:, s_k+1:end)$
13:         $\Sigma_k^{(1)} = \Sigma_k(1:s_k, 1:s_k)$
14:         $V_k^{(1)} = V_k(:, 1:s_k)$
15:         $P_k^{(2)} = P_k U_k^{(2)}$
16:         $H = [H, P_k^{(2)}]$
17:         $\alpha_k = \Sigma_k^{(1)} {V_k^{(1)}}^\top$
18:         $P_k = P_k U_k^{(1)}$
19:     end if
20:     $X_k = X_{k-1} + P_k \alpha_k$
21:     $R_k = R_{k-1} - AP_k \alpha_k$
22:     $P_{k+1} = AP_k - P_k P_k^\top AAP_k - P_{k-1} P_{k-1}^\top AAP_k - HH^\top AAP_k$
23:     $P_{k+1} =$ A-orthonormalize($P_{k+1}$)
24:     $k = k + 1$
25: end while
```

4 LORASC Preconditioner

In this section we introduce LORASC, a robust algebraic preconditioner of the form $M = (L + \tilde{D})\tilde{D}^{-1}(\tilde{D} + L^T)$ that can be efficiently built and applied in parallel. Part of the presentation of LORASC corresponds to the material published in [4]. The graph of the input matrix A is first partitioned by using k-way partitioning with vertex separators into N disjoint domains and a separator formed by the vertices connecting the N domains. Such a partitioning can be obtained by using existing software as METIS [8]. The permuted matrix has a block arrow structure, as presented in equation (4.1), in which the first N diagonal blocks correspond to the disjoint domains, while the last diagonal block $A_{\Gamma\Gamma}$ corresponds to the separator.

$$A = \begin{pmatrix} A_{11} & & & A_{1\Gamma} \\ & \ddots & & \vdots \\ & & A_{NN} & A_{N\Gamma} \\ A_{\Gamma 1} & \cdots & A_{\Gamma N} & A_{\Gamma\Gamma} \end{pmatrix}. \tag{4.1}$$

LORASC is algebraic in the sense that no information from the underlying PDE is required, neither for the construction of the preconditioner, nor for the graph partitioning method used for parallelism. It is robust in the sense that the spectral condition number (defined as the ratio of the largest to the smallest eigenvalue) of the preconditioned matrix $\kappa(M^{-1}A)$ is bounded by a user defined value τ. Here both M and A are SPD matrices. The preconditioner relies on the Cholesky factorization of the first N diagonal blocks and on approximating the Schur complement $S = A_{\Gamma\Gamma} - \sum_{j=1}^{N} A_{\Gamma j} A_{jj}^{-1} A_{j\Gamma}$ which would be computed in a direct factorization of A (however prohibitive for 3D large problems). Our preconditioner is obtained by approximating first the inverse of S by $A_{\Gamma\Gamma}^{-1}$. With this approximation, the eigenvalues of $A_{\Gamma\Gamma}^{-1}S$ are upper bounded by 1. The approximation of the Schur complement $\tilde{S}^{-1}$ is obtained by correcting the first approximation $A_{\Gamma\Gamma}^{-1}$ by a low rank matrix which allows to shift all the eigenvalues of $A_{\Gamma\Gamma}^{-1}S$ smaller than a threshold $\varepsilon = 1/\tau$ to ε. We use for this a technique inspired from Wielandt's deflation (see [14]). The condition number of $\tilde{S}^{-1}S$ and also of the overall preconditioned matrix $M^{-1}A$ is bounded by τ. The smallest eigenvalues and associated eigenvectors of $A_{\Gamma\Gamma}^{-1}S$ are computed by solving a generalized eigenvalues problem of the form

$$Su = \lambda A_{\Gamma\Gamma} u, \;\; S = A_{\Gamma\Gamma} - \sum_{j=1}^{N} A_{\Gamma j} A_{jj}^{-1} A_{j\Gamma}. \tag{4.2}$$

In our experiments we observe that the size of the low rank correction increases slower than linearly with the number of domains. This is an important property, since the number of domains corresponds to the number of processors that will be used for the parallel execution of LORASC. This means that the time required to apply the preconditioner will increase slowly when increasing the number of processors, and this will allow to obtain a scalable application of LORASC during the iterative process.

Given a matrix A of size $n \times n$, we refer to its spectrum $\Lambda(A)$ as

$$\Lambda(A) = \{\lambda_1, \lambda_2, ..., \lambda_n\},$$

where $\lambda_1 = \lambda_{min}(A)$ is its smallest eigenvalue and $\lambda_n = \lambda_{max}(A)$ is its largest eigenvalue.

In the following we consider that the input matrix A has been reordered by using k-way graph partitioning with vertex separators. The obtained matrix A has a bordered block diagonal form, as presented in equation (4.1), also referred to as block arrow matrix, where each diagonal block corresponds to a domain, while the last diagonal block $A_{\Gamma\Gamma}$ corresponds to the separator, the frontier between domains. The block diagonal matrices A_{ii} are of dimension $n_i \times n_i$, for $i = 1, \ldots, N$, and the last diagonal block $A_{\Gamma\Gamma}$ is of dimension $n_\Gamma \times n_\Gamma$.

The matrix A can be factored as

$$A = \begin{pmatrix} A_{11} & & & \\ & \ddots & & \\ & & A_{NN} & \\ A_{\Gamma 1} & \cdots & A_{\Gamma N} & S \end{pmatrix} \begin{pmatrix} A_{11}^{-1} & & & \\ & \ddots & & \\ & & A_{NN}^{-1} & \\ & & & S^{-1} \end{pmatrix} \begin{pmatrix} A_{11} & & & A_{1\Gamma} \\ & \ddots & & \vdots \\ & & A_{NN} & A_{N\Gamma} \\ & & & S \end{pmatrix}, \tag{4.3}$$

where the Schur complement S is computed as

$$S = A_{\Gamma\Gamma} - \sum_{j=1}^{N} A_{\Gamma j} A_{jj}^{-1} A_{j\Gamma}. \tag{4.4}$$

Consequently, the factorization of A can be written as

$$A = (L+D)D^{-1}(D+L^T), \tag{4.5}$$

with $D = \text{Block-Diag}(A_{11}, A_{22}, ..., A_{NN}, S)$ and

$$L = \begin{pmatrix} 0 & & & \\ & \ddots & & \\ & & 0 & \\ A_{\Gamma 1} & \cdots & A_{\Gamma N} & 0 \end{pmatrix}, \tag{4.6}$$

Since for problems arising from the discretization of PDEs on large 3D grids, the Schur complement S becomes fairly dense, direct methods of factorization are prohibitive in terms of memory and computation costs. In our preconditioner we approximate S by a much sparser matrix $\tilde{S}$.

Consider a symmetric positive definite matrix A of size $n \times n$, which has a bordered block diagonal structure as in Equation (4.1). We refer in the following to the Schur complement preconditioner as M, and the preconditioned linear system that we solve is

$$M^{-1}Ax = M^{-1}b \tag{4.7}$$

The preconditioner M is defined by the following approximate factorization

$$\begin{aligned} M &= (L+\tilde{D})\tilde{D}^{-1}(\tilde{D}+L^T) \qquad (4.8) \\ &= \begin{pmatrix} A_{11} & & & \\ & \ddots & & \\ & & A_{NN} & \\ A_{\Gamma 1} & \cdots & A_{\Gamma N} & \tilde{S} \end{pmatrix} \begin{pmatrix} A_{11}^{-1} & & & \\ & \ddots & & \\ & & A_{NN}^{-1} & \\ & & & \tilde{S}^{-1} \end{pmatrix} \begin{pmatrix} A_{11} & & & A_{1\Gamma} \\ & \ddots & & \vdots \\ & & A_{NN} & A_{N\Gamma} \\ & & & \tilde{S} \end{pmatrix}, \end{aligned}$$

where $\tilde{D} = \text{Block-Diag}(A_{11}, A_{22}, ..., A_{NN}, \tilde{S})$, L is defined as in Equation (4.6), and $\tilde{S}$ is an approximation of the Schur complement S from Equation (4.4).

It can be easily shown that $\Lambda(M^{-1}A) = \Lambda(\tilde{S}^{-1}S) \cup \{1\}$ (see in [4]).

We start from the observation that $\lambda_{max}(A_{\Gamma\Gamma}^{-1}S) \leq 1$. This is because

$$A_{\Gamma\Gamma}^{-1}S = I - A_{\Gamma\Gamma}^{-1}C, \text{ with } C := \sum_{j=1}^{N} A_{\Gamma j} A_{jj}^{-1} A_{j\Gamma}, \tag{4.9}$$

and $\Lambda(A_{\Gamma\Gamma}^{-1}C) = \Lambda(A_{\Gamma\Gamma}^{-\frac{1}{2}} C A_{\Gamma\Gamma}^{-\frac{1}{2}})$. Then the fact that $\lambda_{min}(A_{\Gamma\Gamma}^{-\frac{1}{2}} C A_{\Gamma\Gamma}^{-\frac{1}{2}}) \geq 0$ gives $\lambda_{max}(A_{\Gamma\Gamma}^{-1}S) \leq 1$.

However, a preconditioner based only on $A_{\Gamma\Gamma}$ does not allow to lower bound the eigenvalues of $\tilde{S}^{-1}S$. To solve this problem, we use a formulation of $\tilde{S}^{-1}$ that adds to $A_{\Gamma\Gamma}^{-1}$ a low rank matrix allowing to correct the smallest eigenvalues of $A_{\Gamma\Gamma}^{-1}S$. Our deflation method is inspired from the Wiedlandt's deflation technique explained in [14]. The correction matrix shifts the smallest eigenvalues of $A_{\Gamma\Gamma}^{-1}S$ to a prescribed lower bound $\varepsilon = \frac{1}{\tau}$. Note that since the application of M during the iterative process requires applying the inverse of $\tilde{S}$, in the following we discuss the formulation of $\tilde{S}^{-1}$ rather than the formulation of $\tilde{S}$.

In more details, we fix a threshold τ for the required spectral condition number $\kappa(\tilde{S}^{-1}S)$ which leads us to prescribe a lower bound $\varepsilon = \frac{1}{\tau}$ for the eigenvalues of $\tilde{S}^{-1}S$, as we know that $\lambda_{max}(A_{\Gamma\Gamma}^{-1}S) \leq 1$. We use the generalized eigenvalues problem

$$Su = \lambda A_{\Gamma\Gamma}u. \tag{4.10}$$

Let $\lambda_1, \lambda_2, ..., \lambda_i$ be the generalized eigenvalues that need to be corrected, i.e. $\lambda_k < \varepsilon, k \in \{1, 2, ..., i\}$, and let $v_1, v_2, ..., v_i$ be the corresponding $A_{\Gamma\Gamma}$-orthonormal generalized eigenvectors (see [9, Theorem 1.11]). The inverse of the approximation $\tilde{S}$ is defined as

$$\tilde{S}^{-1} = A_{\Gamma\Gamma}^{-1} + E_i \Sigma_i E_i^T. \tag{4.11}$$

where $E_i = (v_1 \quad v_2 \quad ... \quad v_i)$ and Σ_i is defined as

$$\Sigma_i = \mathrm{Diag}(\sigma_1, \sigma_2, \ldots, \sigma_i) \tag{4.12}$$

with $\sigma_1, \sigma_2, ..., \sigma_i$ chosen as

$$\sigma_k = \frac{\varepsilon - \lambda_k}{\lambda_k}, \qquad k \in \{1, 2, ..., i\}. \tag{4.13}$$

Then, one can easily prove that

$$\varepsilon \leq \lambda(\tilde{S}^{-1}S) \leq 1.$$

Hence the constructed matrix $\tilde{S}$ is a good approximation of S which ensures that the spectral condition number $\kappa(\tilde{S}^{-1}S)$ is bounded by a given tolerance $\tau = \frac{1}{\varepsilon}$.

Definition 1 (LORASC preconditioner). *Let A be an $n \times n$ symmetric positive definite matrix with a bordered block diagonal structure,*

$$A = \begin{pmatrix} A_{11} & & & A_{1\Gamma} \\ & \ddots & & \vdots \\ & & A_{NN} & A_{N\Gamma} \\ A_{\Gamma 1} & \cdots & A_{\Gamma N} & A_{\Gamma\Gamma} \end{pmatrix}. \tag{4.14}$$

Let $S = A_{\Gamma\Gamma} - \sum_{j=1}^{N} A_{\Gamma j} A_{jj}^{-1} A_{j\Gamma}$. Given a tolerance τ, a lower bound $\varepsilon = \frac{1}{\tau}$ for the generalized eigenvalues problem $Su = \lambda A_{\Gamma\Gamma}u$, let $\lambda_1, \lambda_2, ..., \lambda_i$ be the generalized eigenvalues smaller than ε, i.e. for all $k \in \{1, ..., i\}$ then $\lambda_k < \varepsilon$, and let $v_1, v_2, ..., v_i$ be the corresponding $A_{\Gamma\Gamma}$-orthonormal generalized eigenvectors.

The LORASC *preconditioner of A is defined as*

$$M_{\mathrm{LORASC}} := (L + \tilde{D})\tilde{D}^{-1}(\tilde{D} + L^T).$$

where L is given by (4.6) *and* $\tilde{D} = \mathrm{Block{-}Diag}(A_{11}, A_{22}, ..., A_{NN}, \tilde{S})$. *The matrix $\tilde{S}$ is defined as*

$$\tilde{S}^{-1} = A_{\Gamma\Gamma}^{-1} + E_i \Sigma_i E_i^T, \tag{4.15}$$

where E_i, Σ_i are defined as

$$\begin{aligned} E_i &= (v_1 \quad v_2 \quad ... \quad v_i), & (4.16) \\ \Sigma_i &= \mathrm{Diag}(\sigma_1, \sigma_2, \ldots, \sigma_i), \quad \textit{with } \sigma_k = \frac{\varepsilon - \lambda_k}{\lambda_k},\ k \in \{1, 2, ..., i\}. & (4.17) \end{aligned}$$

The construction of the LORASC preconditioner is completely algebraic, since no information from the underlying PDE is required.

4.1 Multilevel approach

As described in the previous section, the construction of LORASC requires solving the eigenvalue system $Su = \lambda A_{\Gamma\Gamma}u$. For this solve, using iterative methods, such as ARPACK, requires at each iteration of the method to perform a matrix-vector product $OP \times v$, where $OP = A_{\Gamma\Gamma}^{-1}S$ is the operator and v, a vector of the Krylov subspace. Since this operation requires again $A_{\Gamma\Gamma}^{-1}$, we can recursively call LORASC to solve the system on $A_{\Gamma\Gamma}$. This approach has several advantages with respect to our default approach which uses Cholesky to factorize $A_{\Gamma\Gamma}$. First, this recursive decomposition allows us to take advantage of the machines with hierarchical memories, e.g clusters of multicores, where the top level of LORASC can use MPI processors over the nodes, and the next level can use threads or openmp over the cores of each node. Second, for linear systems of equations where $A_{\Gamma\Gamma}$ is large after the partitioning, we can still take advantage of iterative methods. In some cases, the new preconditioned system involving $A_{\Gamma\Gamma}^{-1}$ can be solved more efficiency using iterative methods than direct methods.

5 Experiments

The BUNDLE matrices corresponds to a pressure problem arising when solving this test case with *Code_Saturne*. *Code_Saturne* a CFD software developped in EDF and one of the application targets of NLAFET.

The elasticity matrices have been used in our group at Inria to assess the robustness of our preconditioners and solvers [3, 2, 16, 10], they arise from the linear elasticity problem. In infinitesimal strain theory, it may be written as follows:

$$\operatorname{div}(\sigma(u)) + f = 0 \qquad \text{on } \Omega, \tag{5.1}$$

$$u = u_D \qquad \text{on } \partial\Omega_D, \tag{5.2}$$

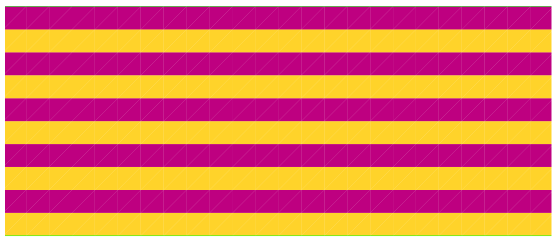
$$\sigma(u)\cdot n = g \qquad \text{on } \partial\Omega_N, \tag{5.3}$$

where Ω be a d-dimensional polygonal or polyhedral domain ($d = 2$ or 3), $u \in \mathbb{R}^d$ is the unknown displacement field, the Dirichlet boundaries $\partial\Omega_D = \{(x,y,z) \in \partial\Omega, x = 0\}$ and the remaining boundaries $\partial\Omega_N$ are the Neumann boundaries, and f is some body force. The Cauchy stress tensor $\sigma(u)$ is given by Hooke's law $\sigma(u) = 2\mu\varepsilon(u) + \lambda \mathrm{Tr}(\varepsilon(u))I$, where Tr is the trace, μ, λ are the Lamé parameters, and are a property of the elastic material, and $\varepsilon(u) = \frac{1}{2}(\nabla u + \nabla u^T)$ is the strain tensor. Note that μ and λ can be expressed in terms of Young's modulus E and Poisson's ratio ν as

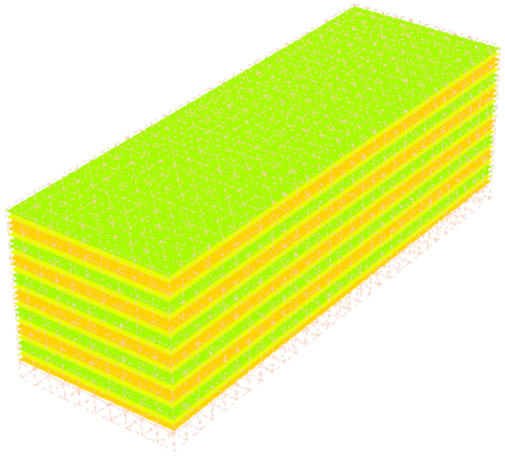
$$\lambda = \frac{\nu E}{(1+\nu)(1-2\nu)}, \qquad \mu = \frac{E}{2(1+\nu)}.$$

The numerical efficiency and robustness of our preconditioner is tested for the two- and three-dimensional systems of linear elasticity on a rectangular and parallelepiped domain, respectively. The domains are discretized with a triangular mesh and $\mathbb{P}_1$ finite elements. The Young's modulus E and Poisson's ratio ν take two values, $(E_1, \nu_1) = (2 \cdot 10^{11}, 0.25)$, and $(E_2, \nu_2) = (10^7, 0.45)$, the distribution is shown in Figure 1.

As stressed in the introduction, we are mainly interested in lowering the iteration count because it corresponds to the number of global synchronizsations.



(a) 2D case



(b) 3D case

Figure 1: The distribution of the Young's modulus.

5.1 Enlarged Conjugate Gradient

All the results are obtained with Matlab R2015b. PCG is the Matlab Preconditioned Conjugate Gradient method. We always use a splitted block Jacobi preconditioner.

We compare BCG [11] and ECG [1] on two set of matrices coming from *Code_Saturne* and linear elasticity problem. These matrices are displayed in Table 1 where we present their size, the number of nonzeros, their smallest and largest eigenvalues.

	Size	Nonzeros	λ_1	λ_n
BUNDLE16	16 384	109 184	8.7e-10	9.8e-06
BUNDLE65	65 536	439 552	2.2e-10	9.8e-06
BUNDLE262	262 144	1 763 840	5.5e-11	9.8e-06
ELA25	9 438	312 372	2.7e-5	3.4
ELA50	18 153	618 747	1.9e-6	3.4
ELA100	36 663	1 231 497	2.6e-7	2.4

Table 1: Matrices used in our tests, their size, number of nonzeros, and smallest (λ_1) and largest (λ_n) eigenvalues.

In Table 2 are summarized the results obtained when increasing the number of right-hand sides with ECG. PCG is Matlab Preconditioned Conjugate Gradient. There is no adaptive reduction of the search directions. The tolerance ε is chosen equal to 10^{-6} and the number of blocks in the preconditioner is equal to 1024. As expected, both on BUNDLE and ELA matrices the number of iterations decreases when the number of right-hand sides increases. On BUNDLE262 matrix, ECG(4) performs almost 2 times less iterations than PCG and about 3 times more than ECG(32). On ELA100 matrix, ECG(4) performs more than 3 times less iterations than PCG and 3 times more than ECG(32).

Matrix	PCG	ECG(4)	ECG(8)	ECG(16)	ECG(32)
BUNDLE16	172	103	79	63	53
BUNDLE65	207	117	92	66	53
BUNDLE262	325	180	127	93	70
ELA25	601	289	214	152	116
ELA50	1001	414	290	208	150
ELA100	955	308	236	155	109

Table 2: Number of iterations to get the solution. The method stops when the relative residual is lower than 10^{-6}. PCG is the usual preconditioned Conjugate Gradient method and ECG(t) is the Enlarged Krylov Conjugate Gradient method where t is the enlarging factor, *i.e.* the number of columns of the initial enlarged residual $T(r_0)$. The system is preconditioned with a block diagonal preconditioner with 1024 blocks. When increasing the enlarging factor the number of iterations decreases, half the number of iterations for ECG(32) compared to ECG(4) and around 4 times less compared to PCG.

In Figure 2, we plot the error (left), as well as the size of the block (right), as a function of the number of iterations when running ECG and BCG with the adaptive reduction of the search directions. The exact solution is computed by using Matlab \. The method stops when the relative residual is lower than 10^{-6} and ε_{def} is chosen equal to $\varepsilon_{\text{def}} = \frac{1}{\sqrt{t}}\varepsilon_{\text{solver}}||b||_2$ as in [2]. The initial block size is 32 and the number of blocks in the preconditioner is 1024. For both matrices, we observe that the convergence of the error is far better for the block methods compared to PCG even with the block size reduction (a factor of 4 for BUNDLE65 and 10 for ELA100). Still there is a small plateau before convergence for all the methods which is a well known phenomenon with Krylov methods [5]. The block size reduction and the error behave similarly for both matrices, the block size is reduced when the system is starting to converge (around iteration 45 for BUNDLE65 and iteration 60 for Ela100). For the two test cases, ECG performs better than BCG, it reduces its search directions more than BCG while keeping a similar convergence speed.

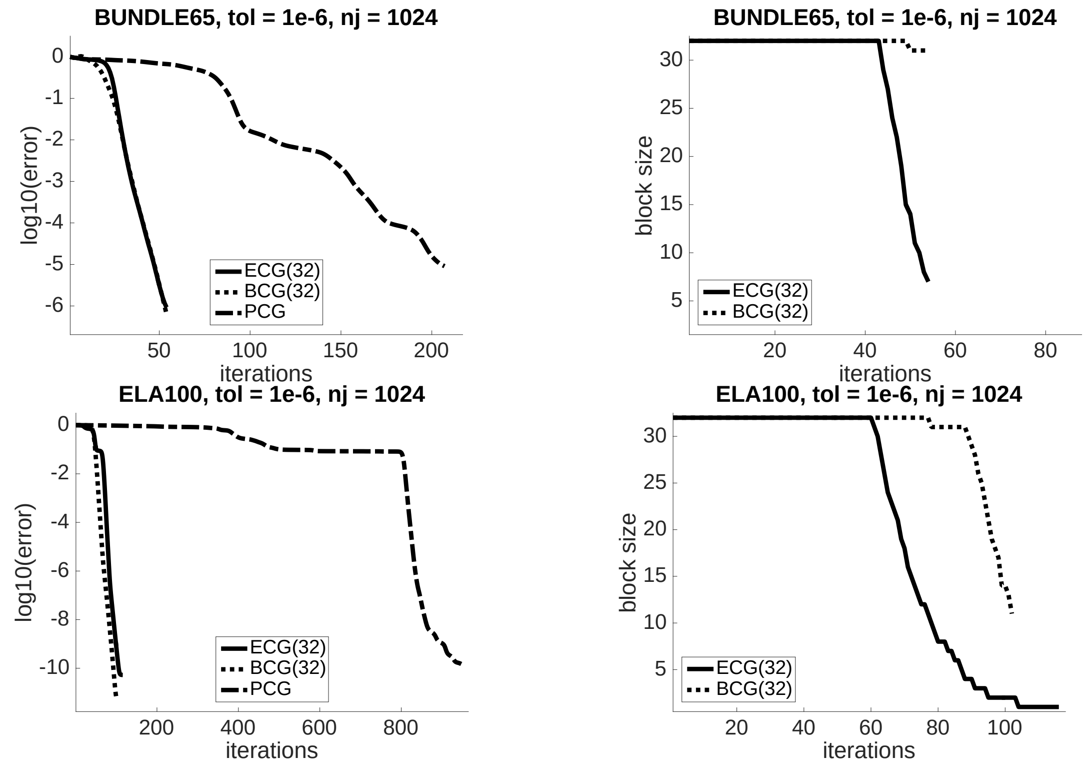


Figure 2: Block size reduction and error decreasing as a function of the number of iterations when using Orthodir with dynamic reduction of the search directions (Algorithm 2). The right figures represent the plot of the error as a function of the number of iterations. We use a log_{10} scale for the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

5.2 LORASC Preconditioner

In this section we analyze the efficiency of the LORASC preconditioner on a set of matrices arising from the discretization by the finite element methods of linear elasticity models, with highly heterogeneous elastic moduli, on three-dimensional (3D) domains. We define the test cases and build the different matrices via FreeFem++ [6], and we build our LORASC preconditioner using MatLab. First the matrices are permuted to a bordered block diagonal form as in equation (4.1) by using the k-way partitioning routine from METIS [8]. Given a number of partitions N, the separators corresponding to the first $\log_2 N$ levels of recursion are grouped together and correspond to $A_{\Gamma\Gamma}$, while the N disjoint subdomains correspond to the first N diagonal blocks.

To solve the linear systems we use CG via MatLab with the threshold 10^{-8} for the stopping criteria of the algebraic iterative resolution. The smallest eigenvalues and associated eigenvectors of the generalized eigenvalues problem are computed using ARPACK.

We refer to the number of nonzero elements as nnz, the number of deflated eigenvalues as n_{EV}. In tables 3, 4 we display N_{mult}, the number of matrix-vector multiplications used by ARPACK to compute the required eigenvalues with a fixed tolerance of 10^{-3}, and $\mathrm{iter}_{\tilde{S}^{-1}}$, the number of iterations of CG preconditioned by LORASC. We also display the number of iterations obtained with a diagonal preconditioner, referred to as $\mathrm{iter}_{A_{\Gamma\Gamma}^{-1}}$. This preconditioner M is obtained once the matrix A has been permuted to a bordered block diagonal form as in equation (4.1). It can be written as $M = \text{Block-Diag}(A_{11}, A_{22}, ..., A_{NN}, A_{\Gamma\Gamma})$ and its application during the iterative process relies on the Cholesky factorization of the diagonal blocks of A. Hence the only difference with LORASC lies in the approximation of the Schur complement S. In the block diagonal preconditioner, S is approximated by $A_{\Gamma\Gamma}$.

As mentioned previously, the smallest eigenvalues and associated eigenvectors of the generalized eigenvalues problem are computed using ARPACK with a fixed tolerance of 10^{-3}. The numerical results focus on testing the behavior of the LORASC preconditioner in terms of weak and strong scalability. In the weak scaling experiments, the size of the problem increases proportionally with the partitions number. In the strong scaling experiments, the problem size is fixed while the number of partitions increases.

5.2.1 Three-dimensional test case

We start by illustrating the behavior of the LORASC preconditioner in terms of weak scaling. Table 3 displays the number of deflated eigenvalues and the number of matrix-vector multiplications performed by ARPACK for the two different thresholds $\tau = 10^2, \tau = 2 \cdot 10^2$. The number of matrix-vector multiplications required in the case of $\varepsilon = 10^{-2}$ and $\varepsilon = 2 \cdot 10^{-3}$ increases slightly. Overall an important gain is obtained by LORASC with respect to the block diagonal preconditioner that uses $A^{-1}_{\Gamma,\Gamma}$ only to approximate $\tilde{S}^{-1}$.

n	N	nnz	$\tilde{S}^{-1}, \varepsilon = 0.01$			$\tilde{S}^{-1}, \varepsilon = 0.005$			$A^{-1}_{\Gamma,\Gamma}$
			n_{EV}	N_{mult}	$\text{iter}_{\tilde{S}^{-1}}$	n_{EV}	N_{mult}	$\text{iter}_{\tilde{S}^{-1}}$	$\text{iter}_{A^{-1}_{\Gamma,\Gamma}}$
4719	2	153057	0	0	71	0	0	71	71
9438	4	312372	5	92	65	3	83	89	113
18513	8	618747	10	111	63	8	95	84	207
36663	16	1231497	15	132	60	11	111	76	267
72963	32	2456997	42	325	55	24	230	64	592

Table 3: Weak scaling results for 3D test cases. The results for LORASC are given in the columns $\tilde{S}^{-1}$ for two values of the parameter ε. The results for the block diagonal preconditioner are given in the column $A^{-1}_{\Gamma,\Gamma}$.

Strong scaling results are presented in Table 4. We note that the size of the deflation space and the number of matrix-vector multiplications performed by ARPACK increases slightly for the two choices of the parameter ε. When comparing the number of matrix-vector multiplications needed to deflate the required smallest eigenvalues plus the number of iterations of LORASC (N_{mult}+ $\text{iter}_{\tilde{S}^{-1}}$), with the number of iterations of the block diagonal preconditioner ($\text{iter}_{A^{-1}_{\Gamma,\Gamma}}$), we observe that LORASC outperforms the diagonal block precondiitoner. The gain obtained by LORASC is more important when the partition number increases, and a factor of roughly 2 is obtained for $\varepsilon = 5 \cdot 10^{-3}$.

Finally, we mention that similar results with the 2D case are obtained when we use more rough ($\varepsilon = 10^{-1}$) or smooth ($\varepsilon = 10^{-3}$) lower bound, for both weak and strong scaling. That is, no significant gain is obtained by LORASC with respect to the block diagonal preconditioner.

Table 5 compares the number of nonzeros in LORASC with respect to the number of nonzeros in the Cholesky factor of the matrix A. This is a 3D case, and the underlying discretization grid has dimensions $200 \times 10 \times 10$. The matrix A is of dimension 72963×72963 and the number of nonzeros of A is $nnz(A) = 2456997$. The number of partitions N varies from 8 to 64. The fourth column gives the number of nonzeros in the Cholesky factor of A. This number changes slightly for different partitions numbers since the permutation returned by Metis can be different. For LORASC we display in the fifth column the average number of nonzeros of

	$\tilde{S}^{-1}, \varepsilon = 0.01$			$\tilde{S}^{-1}, \varepsilon = 0.005$			$A_{\Gamma,\Gamma}^{-1}$
N	n_{EV}	N_{mult}	$\text{iter}_{\tilde{S}^{-1}}$	n_{EV}	N_{mult}	$\text{iter}_{\tilde{S}^{-1}}$	$\text{iter}_{A_{\Gamma,\Gamma}^{-1}}$
2	6	83	58	4	83	74	87
4	13	119	65	8	110	75	168
8	23	202	61	13	119	74	322
16	32	249	61	18	159	69	465
32	42	325	55	24	230	64	592

Table 4: Strong scaling results for 3D test cases for $n = 72963$, and $nnz = 2456997$. The results for LORASC are given in the columns $\tilde{S}^{-1}$ for two values of the parameter ε. The results for the block diagonal preconditioner are given in the column $A_{\Gamma,\Gamma}^{-1}$.

the Cholesky factors of the first N diagonal blocks. The last column displays the number of nonzeros of the Cholesky factor of $A_{\Gamma\Gamma}$. We note that the number of nonzeros in the Cholesky factor of $A_{\Gamma\Gamma}$ increases by a factor slightly bigger than 2 when the number of partitions is doubled. We note that here we simply use nested dissection from Metis to permute the matrix A to a bordered block diagonal form, any other ordering can be used.

				LORASC	
			$A = LL^T$	$A_{ii} = L_{ii}Lii^T, i = 1, \ldots, N$	$A_{\Gamma\Gamma} = L_{\Gamma\Gamma}L_{\Gamma\Gamma}^T$
n	N	nnz(A)	$nnz(L)$	$\sum_{i=1}^{N} nnz(L_{ii})/N$	$nnz(L_{\Gamma\Gamma})$
72963	8	2.4e+06	$25.2e+06$	2.2e+06	$2.3e+05$
72963	16	2.4e+06	$25.4e+06$	8.2e+05	$4.8e+05$
72963	32	2.4e+06	$25.3e+06$	2.3e+05	$1.2e+06$
72963	64	2.4e+06	$25.3e+06$	7.2e+04	$2.8e+06$
72963	128	2.4e+06	$25.4e+06$	6.1+04	$5.1e+06$

Table 5: Comparison of the number of nonzeros of the Cholesky factor of A with respect to the Cholesky factors of the diagonal blocks of A

5.2.2 Multilevel results

Table 6 shows the results of the multilevel formulation of LORASC for a large elasticity 3D problem of size 1161963×1161963, with 39221997 nonzeros entries. The number of partitions N varies from 8 to 128. We denote by LORASC (Level 1), the first call of LORASC, that is, for solving the global system, and LORASC (Level 2), the second call of LORASC, that is, for solving the system involving $A_{\Gamma,\Gamma}^{-1}$.
For LORASC (Level 1) and LORASC (Level 2), we use $\varepsilon = 0.01$, and we solve the system with the precision 10^{-8}. The column *PCG* shows the results of Block Jacobi for solving the global system. The column $\text{iter}_{A_{\Gamma,\Gamma}^{-1}}$ shows the average number of iterations required to solve the system with $A_{\Gamma,\Gamma}$, while the column $\text{iter}_{A^{-1}}$ shows the number of iterations to solve the global system with A.
We observe that all the computed eigenvalues of $A_{\Gamma,\Gamma}$ at the second level are greater than $\varepsilon = 0.01$, hence none of them are selected with our deflation technique. Indeed, for the second level, any other preconditioner such as Block Jacobi (PCG) could be used. In our experiments, we found that solving the system at the second level using PCG leads to the same number of iterations as using LORASC at that level. We note that, on contrary to the elasticity problem,

some large problems might lead to an ill-conditioned matrix $A_{\Gamma,\Gamma}$ for which, using deflation technique such as LORASC will still be required to achieve fast convergence.
We note that the number of iterations of LORASC decreases considerably when the number of processors increases. The multilevel version of LORASC outperforms Block Jacobi in terms of iterations. For example, for $N = 128$, Solving the system using LORASC requires 95 iterations, while using Block Jacobi preconditioner requires 18899 iterations. We also observe that the number of matrix-vector products required to build LORASC (Level 1) increases with the number of processors, this is due to the increasing number of eigenvalues to deflate. However, these matrix-vector products are more suitable for parallelism.

N	LORASC (Level 2)			LORASC (Level 1)			PCG
	n_{EV}	N_{mult}	$\text{iter}_{A^{-1}_{\Gamma,\Gamma}}$	n_{EV}	N_{mult}	$\text{iter}_{A^{-1}}$	$\text{iter}_{A^{-1}}$
8	0	35	5	25	4424	247	3747
16	0	57	4	54	5233	159	5861
32	0	69	6	113	8085	120	9704
64	0	173	5	231	15055	98	13408
128	0	769	5	454	28880	95	18899

Table 6: Strong scaling results for 3D test cases for $n = 1161963$, and $nnz = 39221997$. The results for LORASC are given for two levels in the columns *Level*1 and *Level*2. The results for the block diagonal preconditioner are given in the column PCG.

6 Conclusion

In this deliverable, we introduced some new algorithms for solving a symmetric positive definite linear system. On massively parallel machines, communication is the performance bottleneck. That is why those algorithms are designed in order to reduce the communication in parallel.

On one hand, we explained a method to reduce the number of search directions adaptively during the iterations of Enlarged Conjugate Gradient (ECG) [1, 2]. This method is cheap because its arithmetic overhead is low, just a SVD decomposition on a small matrix, and it does not need any communication. It is also completely algebraic and could be applied to any block Krylov Conjugate Gradient variant. On the other hand, we detailed the construction and application of LORASC preconditioner [3], a robust and algebraic preconditioner. We have shown how it can be built and applied in parallel. To assess the effectivity of these methods, we performed several numerical experiments on a set of matrices including some coming from EDF CFD code *Code_Saturne* which is one of the industrial application of NLAFET (see *Deliverable D5.1: Requirements Analysis*). Numerical results on a set of matrices arising from the discretization by the finite element method of linear elasticity models illustrate the robustness, and show the fast convergence and efficiency of LORASC preconditioner. When testing ECG with or without adaptive reduction of the search directions, the numerical results show that our method allows to reduce the computational cost of ECG while maintaining a convergence rate very close to the one of ECG, and much better than the one of PCG.

To conclude, we introduced two methods that give promising results for solving SPD linear systems in parallel. These methods are currently implemented in parallel and they will be tested on massively parallel machines. Those results and the associated source code will be presented in details in *Deliverable D4.3*. In particular, we will test the implementation on *Code_Saturne* test cases and for the Cosmic Microwave Background problem in astrophysics (see *Deliverable*

5.1: Requirements Analysis). In addition, we will consider using LORASC as a preconditioner for ECG.

7 Acknowledgments

This project is funded from the European Union's Horizon 2020 research and innovation programme under the NLAFET grant agreement No 671633.

References

[1] L. Grigori, S. Moufawad, and F. Nataf. Enlarged Krylov Subspace Conjugate Gradient Methods for Reducing Communication. *SIAM Journal on Scientific Computing*, 37(2):744–773, 2016. Also as INRIA TR 8266.

[2] L. Grigori and O. Tissot. Reducing the communication and computational costs of enlarged krylov subspaces conjugate gradient. Research Report RR-9023, February 2017.

[3] Laura Grigori, Frédéric Nataf, and Soleiman Yousef. Robust algebraic Schur complement preconditioners based on low rank corrections. Research Report RR-8557, July 2014.

[4] Laura Grigori, Frédéric Nataf, and Soleiman Yousef. Robust algebraic Schur complement preconditioners based on low rank corrections. Research Report RR-8557, INRIA, July 2014.

[5] M. H. Gutknecht. Block Krylov space methods for linear systems with multiple right-hand sides: an introduction. *in: Modern Mathematical Models, Methods and Algorithms for Real World Systems (A.H. Siddiqi, I.S. Duff, and O. Christensen, eds.)*, pages 420–447, 2007.

[6] F. Hecht. New development in freefem++. *J. Numer. Math.*, 20(3-4):251–265, 2012.

[7] Magnus R. Hestenes and Eduard Stiefel. Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, 49:409–436, December 1952.

[8] George Karypis and Vipin Kumar. A software package for partitioning unstructured graphs, partitioning meshes, and computing fill-reducing orderings of sparse matrices. *University of Minnesota, Department of Computer Science and Engineering*, 1998.

[9] Massimiliano Lucchesi. *Masonry constructions: mechanical models and numerical applications*, volume 39. Springer, 2008.

[10] Frédéric Nataf, Frédéric Hecht, Pierre Jolivet, and Christophe Prud'Homme. Scalable Domain Decomposition Preconditioners For Heterogeneous Elliptic Problems. In *SC13 - International Conference for High Performance Computing, Networking, Storage and Analysis*, page 11 p., Denver, Colorado, United States, November 2013. ACM.

[11] A. A. Nikishin and A.Yu. Yeremin. Variable block cg algorithms for solving large sparse symmetric positive definite linear systems on parallel computers, i: General iterative scheme. *SIAM J. Matrix Anal. Appl*, 16:1135–1153, 1995.

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

[13] M. Robbé and M. Sadkane. Exact and inexact breakdowns in the block GMRES method. *Linear Algebra Appl.*, 419:265–285, 2006.

[14] Youcef Saad. Numerical solution of large nonsymmetric eigenvalue problems. *Comput. Phys. Comm.*, 53(1-3):71–90, 1989. Practical iterative methods for large scale computations (Minneapolis, MN, 1988).

[15] A. Sluis and H.A. Vorst. The rate of convergence of conjugate gradients. *Numerische Mathematik*, 48(5):543–560, 1986.

[16] N. Spillane, V. Dolean, P. Hauret, F. Nataf, C. Pechstein, and R. Scheichl. Abstract robust coarse spaces for systems of PDEs via generalized eigenproblems in the overlaps. *Numer. Math.*, 126(4):741–770, 2014.