Total and selective reuse of Krylov subspaces for the resolution of sequences of nonlinear structural problems

Pierre Gosselet, Christian Rey, Julien Pebrel

▶ **To cite this version:**

Pierre Gosselet, Christian Rey, Julien Pebrel. Total and selective reuse of Krylov subspaces for the resolution of sequences of nonlinear structural problems. International Journal for Numerical Methods in Engineering, 2013, 94 (1), pp.60-83. 10.1002/nme.4441 . hal-00782841

HAL Id: hal-00782841
https://hal.science/hal-00782841

Submitted on 30 Jan 2013

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Total and selective reuse of Krylov subspaces for the resolution of sequences of nonlinear structural problems

P. Gosselet, C. Rey, J. Pebrel
LMT Cachan, ENS Cachan/CNRS/UPMC/PRES UniverSud Paris
61 Avenue du Président Wilson, 94235 Cachan, France

January 30, 2013

Abstract

This paper deals with the definition and optimization of augmentation spaces for faster convergence of the conjugate gradient method in the resolution of sequences of linear systems. Using advanced convergence results from the literature, we present a procedure based on a selection of relevant approximations of the eigenspaces for extracting, selecting and reusing information from the Krylov subspaces generated by previous solutions in order to accelerate the current iteration. Assessments of the method are proposed in the cases of both linear and nonlinear structural problems.

Keywords: Krylov solvers; multiresolution; model reduction.

1 Introduction

Accelerating the convergence of Krylov iterative solvers [44] is an old issue which has returned to the spotlight because of the increasing number of applications for which these are preferred to direct solvers today. Traditional approaches aim at improving the condition number by using frameworks in which efficient preconditioners exist (e.g. domain decomposition methods [25, 17]), or for which good initialization vectors [19], relevant augmentation subspaces [6, 43, 5] or suitable block strategies (see [1] for a very general block-Lanczos algorithm) are available. For instance:

- For 3D elasticity problems, domain decomposition methods come with "physical" augmentation associated with the global equilibrium of floating substructures (rigid body motions), which makes the methods scalable [13, 30]; for plate and shell problems, additional augmentation through "corner modes" [11, 9, 26] is required.
- For structures with repeated patterns, block strategies are possible [20].

- For restarted algorithms, one can use deflation or augmentation [33, 7], or block techniques [3].
- For problems with multiple right-hand sides, deflation [10, 46, 8] is a rather classical approach.

The problem with these techniques is that they require some *a priori* information which is seldom available, except in specific cases.

Many recent works present theoretical and practical comparisons of the numerous algorithms which have been developed in connection with these ideas [48, 47].

Multiresolution approaches form a general framework in which numerical information is available to accelerate the convergence of Krylov solvers. Multiresolution refers to situations in which the solution of a mechanical problem cannot be achieved through the resolution of a single linear system. For example, calculating the solution of a nonlinear or time-dependent problem or exploring a design of experiment during an optimization procedure requires the resolution of sequences of linear systems. Multiresolution is more general than using multiple right-hand sides because the matrices themselves are likely to change from one system to another (multiple right-hand and left-hand sides). Thus, the problem consists in solving a k-indexed family of large, sparse, linear $n \times n$ systems of the form:

$$\boldsymbol{A}^{(k)} x^{(k)} = b^{(k)} \tag{1}$$

Although different, the systems are assumed to be similar to one another. This similarity can be defined in several ways: in terms of rank, by the fact that $\text{rank}(\boldsymbol{A}^{(k)} - \boldsymbol{A}^{(k-1)}) \ll \text{rank}(\boldsymbol{A}^{(k)})$; or in a spectral sense by the fact that the eigenspaces remain stable from one system to another; or in terms of the Krylov subspaces generated [4]. The first case can be dealt with easily, even with direct solvers, by using the Sherman-Morrison formula; the second case requires augmentation strategies in order to eliminate the most penalizing part of the spectrum and improve the active condition number[1] [39, 16, 38, 50]; and the last case calls for preconditioning techniques [40, 41].

While most of the studies of Krylov methods for multiresolution (often referred to as the recycling of Krylov subspaces) are set in the framework of GMRes/MinRes [38, 50], we chose to work on the specific case of the resolution of symmetric, positive definite systems using conjugate gradients (CGs), in which the convergence is under control and related to easily calculated spectral properties [49]. In earlier works, the authors developed efficient preconditioners based on previous Krylov subspaces [40, 41] which took advantage of the conjugation properties of CGs, but did not extract the most interesting part of the information available in the Krylov subspaces, and they proposed augmentation techniques using Ritz vectors [39]. Typically, these works were aimed at nonlinear mechanical systems solved by Newton-Raphson linearization and FETI or BDD domain decomposition [18, 27].

[1] "active" referring to the part of the spectrum of the matrix which is solicited by the right-hand side.

The recycling of Krylov subspaces can also be analyzed from the model reduction point of view. Since Krylov solvers satisfy Petrov-Galerkin conditions, they share many common points with strategies based on Karhunen-Loeve expansion [31, 42, 36]. These similarities are well-known [15, 14, 21]. But our objective is not to develop reduced models of mechanical systems in order to perform fast but coarse analyzes; it is to define, improve and reuse reduced models in order to carry out calculations both rapidly and accurately.

In this paper, we undertake a more in-depth investigation of augmentation using a selection of post-processed Ritz vectors. In Section 2, we begin with a detailed presentation of the theoretical framework of the augmented preconditioned conjugate gradient method; then, in Section 3, we propose a first reuse algorithm in a multiresolution framework (TRKS); in Section 4, we improve this algorithm by proposing a procedure for selecting the "best" Ritz vectors (SRKS and "cluster"); finally, in Section 5, we propose an evaluation of the method in the case of nonlinear mechanics and parametric problems, using domain decomposition methods [17] to define efficient preconditioners.

2 The augmented preconditioned conjugate gradient method

2.1 Algorithm and properties

Let us consider the linear problem

$$\boldsymbol{A}x = b, \tag{2}$$

where $\boldsymbol{A}$ is an $n \times n$ symmetric positive definite matrix, and let us study the resolution of this system using the augmented preconditioned conjugate gradient algorithm. With $\boldsymbol{M}$ being the $n \times n$ symmetric positive definite matrix of the preconditioner, we introduce the following notations:

$$\begin{aligned} i = 0 \dots m \qquad & \text{the iteration number} \\ x_i \qquad & \text{the } i^{th} \text{ approximation} \\ r_i = b - Ax_i = A(x - x_i) \qquad & \text{the } i^{th} \text{ residual} \end{aligned} \tag{3}$$

With no loss of generality, the presentation can be limited to the case of a zero initial guess $x_{00} = 0$. (Otherwise, one can set $b \leftarrow b - \boldsymbol{A}x_{00}$.)

Let $\mathcal{C}$ be a subspace of $\mathbb{R}^n$ of dimension n_c, and let Matrix $\boldsymbol{C} = [c_1, \dots, c_{n_c}]$ be a basis of $\mathcal{C}$. The search principle of the augmented left-preconditioned conjugate gradient is:

$$\left\{ \begin{array}{ll} \text{find} & x_i \in \mathcal{K}_i(\boldsymbol{M}^{-1}\boldsymbol{A}, \mathcal{C}, \boldsymbol{M}^{-1}r_0) \\ \text{such that} & r_i \perp \mathcal{K}_i(\boldsymbol{M}^{-1}\boldsymbol{A}, \mathcal{C}, \boldsymbol{M}^{-1}r_0) \end{array} \right. \tag{4}$$

where $\mathcal{K}_i(\boldsymbol{M}^{-1}\boldsymbol{A}, \mathcal{C}, \boldsymbol{M}^{-1}r_0)$ is the augmented Krylov subspace associated with preconditioned operator $\boldsymbol{M}^{-1}\boldsymbol{A}$ and augmentation subspace $\mathcal{C}$:

$$\mathcal{K}_i(\boldsymbol{M}^{-1}\boldsymbol{A}, \mathcal{C}, \boldsymbol{M}^{-1}r_0) = \text{span}\left(\boldsymbol{M}^{-1}r_0, \dots, (\boldsymbol{M}^{-1}\boldsymbol{A})^{(i-1)}\boldsymbol{M}^{-1}r_0\right) \oplus \mathcal{C} \tag{5}$$

A classical implementation relies on the definition of a convenient initialization and projector pair (x_0,$\boldsymbol{P}$):

$$x = x_0 + \boldsymbol{P}y \quad \left\{ \begin{array}{lll} \boldsymbol{C}^T r_0 = 0 & \Leftarrow & x_0 = \boldsymbol{C}(\boldsymbol{C}^T\boldsymbol{A}\boldsymbol{C})^{-1}\boldsymbol{C}^T b \\ \boldsymbol{C}^T\boldsymbol{A}\boldsymbol{P} = 0 & \Leftarrow & \boldsymbol{P} = \boldsymbol{I} - \boldsymbol{C}(\boldsymbol{C}^T\boldsymbol{A}\boldsymbol{C})^{-1}\boldsymbol{C}^T\boldsymbol{A} \end{array} \right. \tag{6}$$

One should note that since $\boldsymbol{A}\boldsymbol{P} = \boldsymbol{P}^T\boldsymbol{A}\boldsymbol{P} = \boldsymbol{P}^T\boldsymbol{A}$ augmentation preserves symmetry. One should also note that $\boldsymbol{P}\boldsymbol{C} = 0$. The system to be solved is:

$$\boldsymbol{A}\boldsymbol{P}y = (\boldsymbol{P}^T\boldsymbol{A}\boldsymbol{P})y = r_0 = \boldsymbol{P}^T b \tag{7}$$

The $\boldsymbol{C}$-augmented, $\boldsymbol{M}$-preconditioned conjugate gradient technique (APCG) implemented by projection is presented in Algorithm 1. (For the sake of simplicity, the methods will be described assuming exact arithmetic, even though they are compatible with more realistic full reorthogonalization [28].)

Algorithm 1: APCG($\boldsymbol{A}, \boldsymbol{M}, \boldsymbol{C}, b$)

Calculate $\boldsymbol{A}\boldsymbol{C}$, $(\boldsymbol{C}^T\boldsymbol{A}\boldsymbol{C})^{-1}$; $(\boldsymbol{P} = \boldsymbol{I} - \boldsymbol{C}\left(\boldsymbol{C}^T\boldsymbol{A}\boldsymbol{C}\right)^{-1}\boldsymbol{C}^T\boldsymbol{A})$;
$x_0 = \boldsymbol{C}(\boldsymbol{C}^T\boldsymbol{A}\boldsymbol{C})^{-1}\boldsymbol{C}^T b$;
$r_0 = b - \boldsymbol{A}x_0 = \boldsymbol{P}^T b$;
$z_0 = \boldsymbol{P}\boldsymbol{M}^{-1}r_0$, $w_0 = z_0$;
for $j = 1, \ldots, m$ **do**
  $\alpha_{j-1} = (r_{j-1}, w_{j-1})/(\boldsymbol{A}w_{j-1}, w_{j-1})$
  $x_j = x_{j-1} + \alpha_{j-1}w_{j-1}$
  $r_j = r_{j-1} - \alpha_{j-1}\boldsymbol{A}w_{j-1}$
  $z_j = \boldsymbol{P}\boldsymbol{M}^{-1}r_{j+1}$
  $w_j = z_j - \beta_j w_{j-1}$
  $\beta_j = (\boldsymbol{A}w_{j-1}, z_j)/(w_{j-1}, \boldsymbol{A}w_{j-1})$
end

The following basic relations hold:

$$\begin{aligned} (r_i, z_j) &= 0, \quad i \neq j \\ (w_i, \boldsymbol{A}w_j) &= 0, \quad i \neq j \end{aligned} \tag{8}$$

With $\boldsymbol{W}_i = [w_0, \ldots, w_{i-1}]$ and $\boldsymbol{Z}_i = [z_0, \ldots, z_{i-1}]$ being two bases of $\mathcal{K}_i(\boldsymbol{P}\boldsymbol{M}^{-1}\boldsymbol{A}, z_0)$, the projector enables the spaces to be divided orthogonally:

$$\mathcal{K}_i(\boldsymbol{M}^{-1}\boldsymbol{A}, \mathcal{C}, \boldsymbol{M}^{-1}r_0) = \mathcal{K}_i(\boldsymbol{P}\boldsymbol{M}^{-1}\boldsymbol{A}, z_0) \overset{\perp_A}{\oplus} \mathcal{C} \tag{9}$$

Of course, in the absence of optional constraints ($\boldsymbol{C} = 0$, $\boldsymbol{P} = \boldsymbol{I}$), APCG reduces to standard preconditioned conjugate gradients PCG($\boldsymbol{A}$,$\boldsymbol{M}$, b); if, in addition, $\boldsymbol{M}^{-1} = \boldsymbol{I}$, it becomes a standard conjugate gradient algorithm CG($\boldsymbol{A}$, b).

Let us recall a first result which was proven in [6] for the case of non-preconditioned augmented conjugate gradients.

Proposition 1. *Let* $\mathcal{V} = \text{Range}(\boldsymbol{P})$

- *APCG(*$\boldsymbol{A}, \boldsymbol{I}, \boldsymbol{C}$*, b) is equivalent to CG(*$\boldsymbol{P}^T\boldsymbol{A}\boldsymbol{P}_{|\boldsymbol{A}\mathcal{V}}$*,* $\boldsymbol{P}^T b$*) in the sense that both generate the same residuals.* x_i*, the* i^{th} *APCG approximation, is connected to* y_i*, the* i^{th} *CG approximation, by* $x_i = x_0 + \boldsymbol{P}y_i$*.*
- *APCG(*$\boldsymbol{A}, \boldsymbol{I}, \boldsymbol{C}$*, b) does not break down; it converges, and its asymptotic convergence rate is governed by the condition number* $\kappa\left(\boldsymbol{P}^T\boldsymbol{A}\boldsymbol{P}_{|\boldsymbol{A}\mathcal{V}}\right) \leqslant \kappa\left(\boldsymbol{A}\right)$*.*

Consequently, augmentation strategies never decrease the asymptotic convergence rate. The following corollary is straightforward:

Corollary 1. *Let* $\boldsymbol{D} = [d_1, \ldots, d_{m_d}]$ *be a set of* m_d *linearly independent vectors such that* $\boldsymbol{E} = [\boldsymbol{C}, \boldsymbol{D}]$ *is a full column rank matrix. Let* $\boldsymbol{P}_{\boldsymbol{E}} = \boldsymbol{I} - \boldsymbol{E}(\boldsymbol{E}^T\boldsymbol{A}\boldsymbol{E})^{-1}\boldsymbol{E}^T\boldsymbol{A}$*, and let* $\mathcal{V}_{\boldsymbol{E}}$ *be the range of* $\boldsymbol{P}_{\boldsymbol{E}}$*.*

- *APCG(*$\boldsymbol{A}, \boldsymbol{I}, \boldsymbol{E}$*, b) is equivalent to APCG(*$\boldsymbol{P}^T\boldsymbol{A}\boldsymbol{P}$*,* $\boldsymbol{I}$*,* $\boldsymbol{D}$*,* $\boldsymbol{P}^T b$*) in the sense that both generate the same residual.* $x_i^{\boldsymbol{E}}$*, the* i^{th} *approximation of APCG(*$\boldsymbol{A}, \boldsymbol{I}, \boldsymbol{E}$*, b), is connected to* $x_i^{\boldsymbol{D}}$*, the* i^{th} *approximation of APCG(*$\boldsymbol{P}^T\boldsymbol{A}\boldsymbol{P}$*,* $\boldsymbol{I}$*,* $\boldsymbol{D}$*,* $\boldsymbol{P}^T b$*), by* $x_i^{\boldsymbol{E}} = x_0 + \boldsymbol{P}x_i^{\boldsymbol{D}}$*.*
- *The asymptotic convergence rate is governed by* $\kappa\left({\boldsymbol{P}_{\boldsymbol{E}}}^T\boldsymbol{A}\boldsymbol{P}_{\boldsymbol{E}|\boldsymbol{A}\mathcal{V}_{\boldsymbol{E}}}\right) \leqslant \kappa\left(\boldsymbol{P}^T\boldsymbol{A}\boldsymbol{P}_{|\boldsymbol{A}\mathcal{V}}\right) \leqslant \kappa\left(\boldsymbol{A}\right)$*.*

In conclusion, an increase in the size of the augmentation can only improve the asymptotic rate of convergence. (In the worst case, it leaves it unchanged.)

Now let us focus on the effect of preconditioning. Since $\boldsymbol{M}$ is a symmetric positive definite matrix, it can be factorized in Cholesky form $\boldsymbol{M} = \boldsymbol{L}\boldsymbol{L}^T$ (where $\boldsymbol{L}$ denotes a lower triangular matrix with positive diagonal coefficients). Let us introduce the notation:

$$\begin{array}{l} \hat{\boldsymbol{A}} = \boldsymbol{L}^{-1}\boldsymbol{A}\boldsymbol{L}^{-T} \quad ; \quad \hat{b} = \boldsymbol{L}^{-1}b \quad ; \quad \hat{x} = \boldsymbol{L}^T x \\ \hat{\boldsymbol{C}} = \boldsymbol{L}^T\boldsymbol{C} \quad ; \quad \hat{\boldsymbol{P}} = \boldsymbol{I} - \hat{\boldsymbol{C}}(\hat{\boldsymbol{C}}^T\hat{\boldsymbol{A}}\hat{\boldsymbol{C}})^{-1}\hat{\boldsymbol{C}}^T\hat{\boldsymbol{A}} \end{array} \tag{10}$$

Then, the following equivalence between preconditioned and non-preconditioned augmented conjugate gradients holds:

Proposition 2. *APCG(*$\boldsymbol{A}, \boldsymbol{M}, \boldsymbol{C}$*, b) is equivalent to APCG(*$\hat{\boldsymbol{A}}$*,* $\boldsymbol{I}$*,* $\hat{\boldsymbol{C}}$*,* $\hat{b}$*) with* $\hat{r} = \boldsymbol{L}^{-1}r = \hat{z} = \boldsymbol{L}^T z$*,* $\hat{w} = \boldsymbol{L}^T w$*,* $\hat{\alpha} = \alpha$ *and* $\hat{\beta} = \beta$*. Its asymptotic convergence rate is governed by* $\kappa\left(\hat{\boldsymbol{P}}^T\hat{\boldsymbol{A}}\hat{\boldsymbol{P}}_{|\hat{\boldsymbol{A}}\hat{\mathcal{V}}}\right) \leqslant \kappa\left(\hat{\boldsymbol{A}}\right)$*.*

Proof. Since $\hat{\boldsymbol{P}} = \boldsymbol{L}^T\boldsymbol{P}\boldsymbol{L}^{-T}$, we obtain directly $\hat{x_0} = \hat{\boldsymbol{C}}(\hat{\boldsymbol{C}}^T\hat{\boldsymbol{A}}\hat{\boldsymbol{C}})^{-1}\hat{\boldsymbol{C}}^T\hat{b} = \boldsymbol{L}^T x_0$, $\hat{r_0} = \hat{b} - \hat{\boldsymbol{A}}\hat{x_0} = \boldsymbol{L}^{-1}r_0 = \hat{z_0} = \boldsymbol{L}^T z_0$ and $\hat{w_0} = \boldsymbol{L}^T w_0$. By induction, it follows that $\hat{\alpha}_{j-1} = (\hat{r}_{j-1}, \hat{z}_{j-1})/(\hat{\boldsymbol{A}}\hat{w}_{j-1}, \hat{w}_{j-1}) = \alpha_{j-1}$, $\hat{r_j} = \boldsymbol{L}^{-1}r_j$, $\hat{\beta}_j = (\hat{\boldsymbol{A}}\hat{w}_{j-1}, \hat{z}_j)/(\hat{\boldsymbol{A}}\hat{w}_{j-1}, \hat{w}_{j-1}) = \beta_j$ and $\hat{w_j} = \boldsymbol{L}^T w_j$. Proposition 1 provides the inequality concerning the asymptotic convergence rate. □

Putting these propositions together, APCG($\boldsymbol{A}, \boldsymbol{M}, \boldsymbol{C}$, b) is equivalent to CG($\boldsymbol{L}^{-1}\boldsymbol{P}^T\boldsymbol{A}\boldsymbol{P}\boldsymbol{L}^{-T}$, $\boldsymbol{L}^{-1}\boldsymbol{P}^T b$). All these results lead us to propose an efficient augmentation by analogy with an equivalent, simpler system solved by classical conjugate gradients.

2.2 Interpretation and choice of the augmentation

From a "constraint" point of view, the projection guarantees the $\boldsymbol{C}$-orthogonality of the residual throughout the iterations ($\boldsymbol{C}^T r_j = 0$). For example, in the FETI domain decomposition method, the residual is the displacement jump between the subdomains; in the case of shell and plate problems, matrix $\boldsymbol{C}$ is introduced to enforce the continuity of the displacement at the corner points [12]. In the BDD domain decomposition method, matrix $\boldsymbol{C}$ is associated with the rigid body motions of floating substructures and, therefore, local Neumann problems in the preconditioner are always well-posed [29]; for shell and plate problems, the matrix is enriched by corner mode corrections [26]. In both cases, matrix $(\boldsymbol{C}^T\boldsymbol{A}\boldsymbol{C})^{-1}$, called a coarse grid matrix, plays a crucial role in the scalability of these methods.

From a "spectral" point of view, augmentation can be used to decrease the active condition number ("active" referring to eigenelements solicited by the right-hand side) and, thus, improve the asymptotic convergence rate. This is called a deflation strategy [8, 5], which boils down to building matrix $\boldsymbol{C}$ by using (approximate) eigenvectors associated with the lowest eigenvalues. Obviously, when $\boldsymbol{C}$ consists of the n_c eigenvectors associated with the lowest eigenvalues ($\lambda_1 \leqslant \ldots \leqslant \lambda_{n_c} \leqslant \ldots \leqslant \lambda_n$), the condition number decreases strictly: $\kappa\left(\boldsymbol{P}^T\boldsymbol{A}\boldsymbol{P}_{\boldsymbol{A}\mathcal{V}}\right) = \frac{\lambda_n}{\lambda_{n_c}} < \frac{\lambda_n}{\lambda_1}$.

From a "model reduction" point of view, subspace $\mathcal{C}$ represents a "macro" (or coarse) space in which the macro part of the solution is calculated directly during the initialization while the "micro" part of the solution, when required, is obtained during the iterations.

2.3 Estimation of computation costs

With regard to the numerical cost of augmentation, the main operations for the construction of the projector are: (i) the block product $\boldsymbol{A}\boldsymbol{C}$ (and assembly with neighbors for domain decomposition methods), (ii) the block dot-product $(\boldsymbol{C}^T\boldsymbol{A}\boldsymbol{C})$ (plus an all-to-all sum for domain decomposition methods), and (iii) the factorization of the fully-populated coarse matrix $(\boldsymbol{C}^T\boldsymbol{A}\boldsymbol{C})$. Then, the application of the projector consists simply of (i) one block dot-product $((\boldsymbol{A}\boldsymbol{C})^T x)$ (plus an all-to-all exchange), (ii) the resolution of the coarse problem, and (iii) the matrix-vector product $(\boldsymbol{C}\alpha)$.

Thus, provided that the number of columns of matrix $\boldsymbol{C}$ is small, the main cost is related to the calculation of $\boldsymbol{A}\boldsymbol{C}$. One must bear in mind that block operations (on "multivectors") are comparatively much faster than single vector operations, especially when the matrices are sparse (because data fetching is factorized). In a domain decomposition context, product $\boldsymbol{A}\boldsymbol{C}$ corresponds to the resolution of Dirichlet or Neumann problems in substructures, which makes the simultaneous treatment of many columns very efficient (and minimizes the number of exchanges). One must also remember that a conjugate gradient iteration involves a preconditioning step which may be expensive. (The cost is comparable to that of an operator product in optimal domain decomposition

methods.) Thus, the additional cost of augmentation relative to the cost of one iteration depends on many parameters (the size of the problem, the number of augmentation vectors, the number of subdomains, the preconditioner chosen...). Typically, in the examples presented in this paper, we found that, using an optimal preconditioner, the CPU cost of between 4 and 7 augmentation vectors (depending on the hardware configuration) cost no more than one CG iteration.

A question which is not addressed in this paper is the verification of the full-rank property of matrix $\boldsymbol{C}$, which affects the quality of the factorization of matrix $(\boldsymbol{C}^T\boldsymbol{A}\boldsymbol{C})$. Strategies to correct a dependence among the columns of matrix $\boldsymbol{C}$ due to inexact arithmetic can be found in [1].

3 Total reuse of Krylov subspaces

In this section, we show how it is possible to define efficient augmentation strategies in a multiresolution context. Let us consider the sequence of linear systems:

$$\boldsymbol{A}^{(k)}x^{(k)} = b^{(k)} \quad , \quad k = 1, \dots p \tag{11}$$

where $A^{(k)}$ is an $n \times n$ symmetric positive definite matrix and $b^{(k)}$ is the right-hand side. Each linear system is solved using an augmented preconditioned conjugate gradient algorithm APCG($\boldsymbol{A}^{(k)}$, $\boldsymbol{M}^{(k)}$, $\boldsymbol{C}^{(k)}$, $b^{(k)}$). Let $m^{(k)}$ be the number of iterations which is necessary to reach convergence, and let

$$\boldsymbol{W}_m^{(k)} = \left[w_0^{(k)}, \dots, w_{m^{(k)}-1}^{(k)}\right] \tag{12}$$

be a basis of the associated Krylov subspace.

As explained in the previous section, augmentation never increases the condition number which governs the asymptotic convergence rate. More precisely, the presence of active eigenvectors of the current preconditioned problem in $\boldsymbol{C}^{(k)}$ may increase the efficiency of the iterative solver significantly. Classical strategies can be used in the case of invariant preconditioned operators ($\boldsymbol{A}^{(k)} = \boldsymbol{A}$, $\boldsymbol{M}^{(k)} = \boldsymbol{M}$) and multiple right-hand sides.

It is more difficult to define efficient strategies in the general case of varying operators with no information available on their evolution. A simple and natural idea is to reuse previous Krylov subspaces. A first algorithm which reuses all the previous Krylov subspaces is Total Reuse of Krylov Subspaces (TRKS) (Algorithm 2), which needs only a few comments:

- Since (according to (8)) ${\boldsymbol{C}^{(k)}}^T\boldsymbol{A}^{(k)}\boldsymbol{W}_m^{(k)} = 0$, the vectors of the concatenated matrix $\boldsymbol{C}^{(k+1)}$ are linearly independent. Therefore, APCG($\boldsymbol{A}^{(k+1)}$, $\boldsymbol{M}^{(k+1)}$, $\boldsymbol{C}^{(k+1)}$, $b^{(k+1)}$) does not break down and converges.

- The previous Krylov subspaces are fully reused through concatenation without post-processing; the only downside is that the memory requirements increase due to the need to save the Krylov subspaces.

- If the number of columns of matrix $\boldsymbol{C}^{(k)}$ becomes too large, the method may become computationally inefficient, even though the number of iterations decreases considerably. Nevertheless, TRKS probably leads to the best reduction in the number of iterations achievable by reusing Krylov subspaces. Therefore, it can be used as a reference in terms of the reduction of the number of iterations for any other algorithm based on a reuse of Krylov subspaces.

- One possible way to reduce the cost of TRKS without reducing the size of $\boldsymbol{C}$ consists in using approximate solvers, as in the IRKS strategy [41].

Algorithm 2: TRKS-APCG

Initialize $\boldsymbol{C}^{(0)} = \boldsymbol{C}_0$ (an $n \times m_0$ full-rank matrix);
for $k = 0, \ldots, p-1$ **do**
  Solve $\boldsymbol{A}^{(k)} x^{(k)} = b^{(k)}$
    with APCG($\boldsymbol{A}^{(k)}$, $\boldsymbol{M}^{(k)}$, $\boldsymbol{C}^{(k)}$, $b^{(k)}$);
  Define $\boldsymbol{W}_m^{(k)} = \left[\ldots, w_j^{(k)}, \ldots\right]_{0 \leqslant j < m^{(k)}}$;
  Concatenate: $\boldsymbol{C}^{(k+1)} = \left[\boldsymbol{C}^{(k)}, \boldsymbol{W}_m^{(k)}\right]$
end

In order to reduce the cost associated with the total reuse of Krylov subspaces, we propose to work on extracted sub-subspaces, an operation often referred to as the recycling of Krylov subspaces. The objective is to retain the smallest number of independent vectors which achieve the greatest decrease in the number of iterations. Clearly, the most effective approach would be to calculate approximate eigenvectors from the previous Krylov subspaces for the current operator. However, because of the variability of the operators, the extraction of such information would be extremely time consuming and would affect the global efficiency. Conversely, approximate eigenvectors of previous problems can be calculated from the associated Krylov subspaces at nearly no cost. In the following section, we describe an efficient algorithm for the extraction of such approximation vectors along with a simple selection procedure to recycle only a few of these vectors. Of course, the performance of our method depends on the stability of the eigenspaces from one system to another. This topic, especially concerning the lower part of the spectrum, is discussed in [24].

4 Selective recycling of Krylov subspaces

The standard convergence of conjugate gradients corresponds to an asymptotic convergence rate. Using this property to predict the number of iterations n_ϵ which is required to reach an accuracy level ϵ_{cg} leads to a huge overestimation.

Indeed, one has:

$$\begin{aligned} &\frac{\|x_i - x\|_{\mathbb{A}}}{\|x_0 - x\|_{\mathbb{A}}} \leqslant 2(\sigma_{1,n})^i \leqslant \epsilon_{cg} \quad \Rightarrow \quad i \geqslant n_\epsilon = \frac{\ln(\epsilon_{cg}/2)}{\ln(\sigma_{1,n})} \\ &\text{with } \sigma_{r,s} = \frac{\sqrt{\kappa_{r,s}} - 1}{\sqrt{\kappa_{r,s}} + 1} \text{ and } \kappa_{r,s} = \frac{\lambda_r}{\lambda_s} \end{aligned} \tag{13}$$

This result alone cannot explain the improvement in the convergence rate observed during the iteration process. This superconvergence phenomenon can be explained by a study of the convergence of Ritz values [33] which enables one to define an instantaneous convergence rate [49]. This explanation can be improved by a study of the influence of the distribution of the eigenvalues [35, 2].

The objective of recycling Krylov subspaces is to find the best augmentation space in order to trigger superconvergence quickly. This section is organized as follows: we start with a review of Ritz eigenelement analysis and continue with a brief presentation of the improved convergence results; these results lead to a number of selection strategies, which will be assessed in Section 5.

4.1 Ritz analysis: theory and practical calculation

For $0 \leqslant i < m$, Ritz vectors $(\hat{y}_m^i)$ and values (θ_m^i) are approximations of the eigenvectors and eigenvalues of the symmetric positive definite matrix $\hat{\boldsymbol{A}}$; their definition is similar to that of the iterates in the conjugate gradient algorithm (4)

$$\left\{ \begin{array}{ll} \text{find} & (\hat{y}_m^i, \theta_m^i) \in \mathcal{K}_m(\hat{\boldsymbol{A}}, \hat{v}_0) \times \mathbb{R} \\ \text{such that} & \hat{\boldsymbol{A}} y_m^i - \theta_m^i y_m^i \perp \mathcal{K}_m(\hat{\boldsymbol{A}}, \hat{v}_0) \end{array} \right. \tag{14}$$

The symmetric Lanczos algorithm [45] enables one to build a particular orthonormal basis of $\mathcal{K}_m(\hat{\boldsymbol{A}}, \hat{v}_0)$, denoted $\boldsymbol{V}_m$. Then, the search principle becomes:

$$y_m^i = \hat{\boldsymbol{V}}_m q_m^i, \quad \hat{\boldsymbol{V}}_m^T \hat{\boldsymbol{A}} \hat{\boldsymbol{V}}_m q_m^i = \theta_m^i q_m^i \tag{15}$$

The Lanczos basis $\hat{\boldsymbol{V}}_m$ makes the Hessenberg matrix $\hat{\boldsymbol{H}}_m = \hat{\boldsymbol{V}}_m^T \hat{\boldsymbol{A}} \hat{\boldsymbol{V}}_m$ symmetrical and tridiagonal. $\hat{\boldsymbol{V}}_m$ and $\hat{\boldsymbol{H}}_m$ can be recovered directly from the conjugate gradient coefficients [44]:

$$\begin{aligned} &\left\{ \begin{array}{l} \hat{\boldsymbol{V}}_m = \left(\ldots, (-1)^j \frac{\hat{r}_j}{\|\hat{r}_j\|}, \ldots \right)_{0 \leqslant j < m} \\ \hat{\boldsymbol{H}}_m = \text{tridiag}(\eta_{j-1}, \delta_j, \eta_j)_{0 \leqslant j < m} \end{array} \right. \\ &\text{with } \delta_0 = \frac{1}{\alpha_0}, \ \delta_j = \frac{1}{\alpha_j} + \frac{\beta_{j-1}}{\alpha_{j-1}}, \ \eta_j = \frac{\sqrt{\beta_j}}{\alpha_j} \end{aligned} \tag{16}$$

Since matrix $\hat{\boldsymbol{H}}_m$ is symmetrical and tridiagonal, its eigenelements $(\theta_j^m, q_j^m)_{1 \leqslant j \leqslant m}$ can be calculated easily, for example using a Lapack procedure. Let us define $\boldsymbol{\Theta}_m = \text{diag}(\theta_m^1 \leqslant \ldots \leqslant \theta_m^m)$ and $\boldsymbol{Q}_m = \left[q_m^1, \ldots, q_m^m\right]$ such that $\hat{\boldsymbol{H}}_m = \boldsymbol{Q}_m \boldsymbol{\Theta}_m {\boldsymbol{Q}_m}^T$. $\boldsymbol{\Theta}_m$ and $\hat{\boldsymbol{Y}}_m = \hat{\boldsymbol{V}}_m \boldsymbol{Q}_m$ are the Ritz values and associated Ritz

vectors, which are approximations of the eigenelements of operator $\hat{\boldsymbol{A}}$ and satisfy:

$$\hat{\boldsymbol{Y}}_m^T \hat{\boldsymbol{A}} \hat{\boldsymbol{Y}}_m = \boldsymbol{\Theta}_m \qquad \text{and} \qquad \hat{\boldsymbol{Y}}_m^T \hat{\boldsymbol{Y}}_m = \boldsymbol{I}_m$$

We presented Ritz analysis for the equivalent symmetric system described previously because symmetry simplifies the calculation of eigenelements, but the analysis can be transferred back to the left-preconditioned system using the following transformation rules:

$$\begin{aligned} \boldsymbol{V}_m &= \boldsymbol{L}^{-T} \hat{\boldsymbol{V}}_m = \left[\ldots, (-1)^j \frac{z_j}{(r_j, z_j)^{1/2}}, \ldots \right] \\ \boldsymbol{H}_m &= \hat{\boldsymbol{H}}_m = \boldsymbol{V}_m^T \boldsymbol{A} \boldsymbol{V}_m \\ \boldsymbol{Y}_m &= \boldsymbol{L}^{-T} \hat{\boldsymbol{Y}}_m = \boldsymbol{V}_m \boldsymbol{Q}_m \end{aligned} \tag{17}$$

The Ritz vectors are the solution of a generalized eigenproblem and satisfy the following orthogonality properties:

$$\boldsymbol{Y}_m^T \boldsymbol{A} \boldsymbol{Y}_m = \boldsymbol{\Theta}_m \qquad \text{and} \qquad \boldsymbol{Y}_m^T \boldsymbol{M} \boldsymbol{Y}_m = \boldsymbol{I}_m \tag{18}$$

One can show that when m increases the Ritz values converge toward the eigenvalues of $\hat{\boldsymbol{A}}$, and that the convergence is either from above or from below depending on their rank [49, 51]:

$$\theta_m^1 \geqslant \theta_{m-1}^1 \geqslant \theta_m^2 \geqslant \ldots \geqslant \theta_m^{m-1} \geqslant \theta_{m-1}^{m-1} \geqslant \theta_m^m \tag{19}$$

In addition, in the case of clearly distinct eigenvalues, the convergence of a Ritz value results in the convergence of the associated Ritz vector.

4.2 Relation between the convergence of conjugate gradients and the convergence of the Ritz values

In [49], the superconvergence phenomenon is explained by the convergence of the Ritz values through the definition, at each iteration, of a instantaneous convergence rate associated with the part of the spectrum that is not yet approximated correctly by the Ritz values: at a given conjugate gradient iteration, one can find a deflated system (with some of its extreme eigenvalues removed) with similar behavior. Let $[\lambda_l, .., \lambda_r]$ be the spectrum of the deflated operator. The equivalent convergence rate is:

$$\|x - x_{i+1}\|_{\mathbb{A}} \leqslant F_{i,l,r}\ 2\ \sigma_{l,r} \|x - x_i\|_{\mathbb{A}} \tag{20}$$

where $F_{i,l,r}$ quantifies the convergence of the l smallest and r largest Ritz values to the extreme eigenvalues:

$$
\begin{aligned}
F_{i,l,r} &= \max_{l'>l} J^{(i)}_{l,l'} \ \max_{r' \geqslant r} L^{(i)}_{r,r'} \\
J^{(i)}_{l,l'} &= \prod_{j=1}^{l} \left| 1 - \frac{\lambda_{l'}}{\lambda_j} \right| \left| 1 - \frac{\lambda_{l'}}{\theta^i_j} \right|^{-1} \\
L^{(i)}_{r,r'} &= \prod_{j=1}^{r} \left| 1 - \frac{\lambda_{n-r'}}{\lambda_{n+1-j}} \right| \left| 1 - \frac{\lambda_{n-r'}}{\theta^i_{i+1-j}} \right|^{-1}
\end{aligned}
$$

Since this result holds for every pair (l, r), the effective convergence rate at Iteration i corresponds to the pair (l, r) which minimizes $\sigma_{i,l,r} = F_{i,l,r}\sigma_{l,r}$.

Then, after some iterations, the superconvergent conjugate gradient algorithm behaves very much like a conjugate gradient algorithm augmented by the extreme eigenvectors which are associated with the converged Ritz values. In a multiresolution context, provided the linear systems have similar spectral properties, the Ritz vectors associated with the converged Ritz values obtained for one system should define a viable augmentation space for the subsequent resolutions.

4.3 Effect of the distribution of the eigenvalues

The effect of the distribution of the eigenvalues on the convergence of conjugate gradients was studied in [35, 2]. The results take into account the fact that preconditioning often leads to clustered eigenvalues as opposed to uniformly distributed eigenvalues, as can be seen in Figure 1.

In addition to other results, the authors showed that if a spectrum consists of p isolated eigenvalues in the high part of the spectrum, p isolated eigenvalues in the low part of the spectrum and $n-2p$ uniformly distributed central eigenvalues, then the conjugate gradient convergence takes the form:

$$
n_\epsilon \geqslant \tilde{n}_\epsilon = 2p + \text{int}\left(\frac{\ln\left(\epsilon_{cg}/2\right)}{\ln\ \sigma_{p+1,n-p}} - \frac{\sum_{i=1}^{p} \ln\left(\frac{\lambda_{n-p+i}}{4\lambda_i} \left(1 - \frac{\lambda_i}{\lambda_{n-p+i}} \right) \right)}{\ln\ \sigma_{p+1,n-p}} \right) \tag{21}
$$

The convergence rate is approximately equal to the classical convergence rate for the central part, plus one iteration per higher eigenvalue and a little more than one iteration per lower eigenvalue. These results can be combined with the work by Jiao [51, 22] on the convergence of Ritz values. In general, since the method is related to the power iteration method, a correct approximation by the Ritz values is obtained first for the highest eigenvalues, then for the lowest part of the spectrum, resulting in superconvergence (which is governed by the asymptotic convergence rate of the reduced spectrum).

4.4 Selection procedures

The results of Section 4.2 lead to a first proposal of a selection procedure for converged Ritz vectors: convergence is identified by the stagnation of the Ritz values; if the conjugate gradient algorithm converges at iteration m, the Ritz values are calculated for the previous two states $\boldsymbol{\Theta}_m$ and $\boldsymbol{\Theta}_{m-1}$. Once ranked, the m most recent Ritz values $\boldsymbol{\Theta}_m$ are compared to the $m-1$ previous values according to the following criteria:

$$\begin{cases} \theta_m^j \text{ has converged if } \dfrac{|\theta_m^j - \theta_{m-1}^j|}{|\theta_m^j|} \leqslant \varepsilon, \qquad 1 \leqslant j \leqslant m-1 \\ \theta_m^{m-j} \text{ has converged if } \dfrac{|\theta_m^{m-j} - \theta_{m-1}^{m-1-j}|}{|\theta_m^{m-j}|} \leqslant \varepsilon, \qquad 0 \leqslant j \leqslant m-2 \end{cases} \tag{22}$$

where ε is a user parameter which is easy to adjust since the criterion is generally either very high (before the convergence of the Ritz value) or very small (after convergence). Figure 1 illustrates that property with the simple example of the operator associated with the decomposition of a linear elastic cube into ten subdomains; in that case, the higher half of the spectrum has converged.

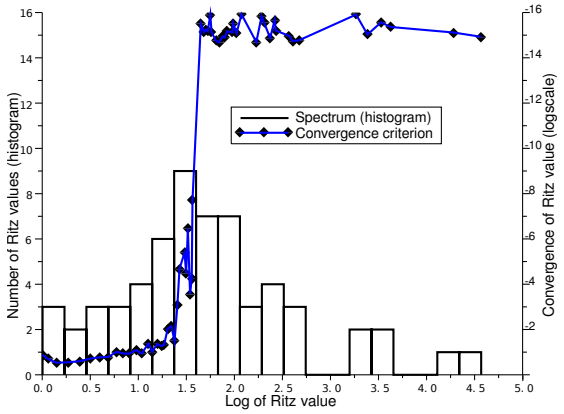

Figure 1: Ritz spectrum and convergence of the Ritz values

The principle of the selective recycling of Krylov subspaces (SRKS-APCG) is described in Algorithm 3. Basically, in addition to the memory required by APCG, the SRKS-APCG algorithm requires storage for m n-vectors $(z_j)_{j=1,m}$. One should note that the selected vectors are normalized by the square root of the associated Ritz value in order to improve the condition number of the coarse matrix. (If operator $\boldsymbol{A}$ remained constant, matrix $(\boldsymbol{C}^T\boldsymbol{A}\boldsymbol{C})$ would be the identity matrix.)

For better computational efficiency, a restart parameter can be introduced in order to limit the size of the augmentation space associated with parameter

Algorithm 3: SRKS-APCG

Initialize $C^{(0)} = \boldsymbol{C}_0$ (full column rank matrix);
for $k = 0, \ldots, p-1$ **do**
- Solve $\boldsymbol{A}^{(k)} x^{(k)} = b^{(k)}$ with APCG$(\boldsymbol{A}^{(k)}, \boldsymbol{M}^{(k)}, \boldsymbol{C}^{(k)}, b^{(k)})$;
- Define $\boldsymbol{V}_m = \left[\ldots, (-1)^j \frac{z_j}{(r_j, z_j)^{1/2}}, \ldots\right]_{0 \leqslant j < m}$;
- Define $\boldsymbol{H}_m = \text{tridiag}(\eta_{j-1}, \delta_j, \eta_j)_{0 \leqslant j < m}$

$\delta_0 = \frac{1}{\alpha_0}$, $\delta_j = \frac{1}{\alpha_j} + \frac{\beta_{j-1}}{\alpha_{j-1}}$, $\eta_j = \frac{\sqrt{\beta_j}}{\alpha_j}$;
- Compute eigenelements $(\boldsymbol{Q}_m, \boldsymbol{\Theta}_m)$ of $\boldsymbol{H}_m$ $(\theta_m^1 \geqslant \ldots \geqslant \theta_m^m)$;
- Compute $\boldsymbol{Y}_m = \boldsymbol{V}_m \boldsymbol{Q}_m = [y_m^1, \ldots, y_m^m]$;
- Extract $\boldsymbol{H}_{m-1} = \text{tridiag}(\eta_{j-1}, \delta_j, \eta_j)_{0 \leqslant j < m-1}$;
- Compute eigenvalues (θ_{m-1}^j) of $\boldsymbol{H}_{m-1}$;

for $j = 1, \ldots, m-1$ **do**
- $\boldsymbol{C} = \left[\boldsymbol{C}, \frac{y_m^j}{\sqrt{|\theta_m^j|}}\right]$ if $|\theta_m^j - \theta_{m-1}^j| \leqslant \varepsilon |\theta_m^j|$;
- $\boldsymbol{C} = \left[\boldsymbol{C}, \frac{y_m^{j+1}}{\sqrt{|\theta_m^{j+1}|}}\right]$ if $|\theta_m^{j+1} - \theta_{m-1}^j| \leqslant \varepsilon |\theta_m^{j+1}|$

end
- Concatenate $\boldsymbol{C}^{(k+1)} = [\boldsymbol{C}^{(k)}; \boldsymbol{C}]$, $\boldsymbol{C} = [0]$;
- If $\dim(\boldsymbol{C}^{(k+1)}) \geqslant n_{c_{lim}}$, then $\boldsymbol{C}^{(k)} = \boldsymbol{C}^{(0)}$

end

$n_{c_{lim}}$ in Algorithm 3. This limit size can be set after a complexity analysis under the assumption that all non-augmented systems would be solved in the same number of iterations. However, we did not use such a restart procedure in our experiments.

In order to be even more selective, we propose a reselection strategy based on a prediction of the efficiency of the retained vectors. Indeed, the results of Section 4.3 in terms of the effect of the distribution of the eigenvalues lead us to retain only the converged Ritz vectors which belong to the external part of the spectrum:

- this is known to be the first part of the spectrum whose approximation by Ritz values is good;
- since the convergence of Ritz vectors is identified by the stagnation of the associated Ritz values, the fact that the external Ritz values are distinct ensures that the Ritz vectors approximate the eigenvectors correctly [51];
- while choosing vectors in the dense central zone does not modify the shape of the spectrum and does not improve convergence, selecting the external part of the spectrum triggers superconvergence instantly.

In order to select only the external part of the spectrum, we implemented the cluster identification algorithm proposed in [32]. This algorithm seeks the

piecewise constant distribution which is nearest (in a least squares sense) to the distribution of the distances among the sorted eigenvalues. The only parameter required is the minimum size of the cluster, which we set at one-fifth the number of preselected vectors. As will be shown in the next section, the performance achieved with this reselection algorithm is not outstanding, but some results in terms of gain per augmentation vector are worth considering.

5 Numerical assessments

We present three numerical experiments. Two concern the evaluation of a structure made of random materials, as is the case in a Monte-Carlo simulation. In the first case, the materials are elastic; in the second case, which is a nonlinear problem, they are elastic-plastic. The last case is a large displacement problem, which raises specific difficulties.

The methods were implemented in the ZEBULON code [34] and parallelism was introduced using MPI. The calculations were performed on the LMT-Cachan cluster, which consists of dual quadcore and dual hexacore processors connected by a gigabit network. The calculations were always carried out on homogeneous sets of processors which were entirely dedicated to one task which fit entirely in memory, so swapping was not necessary. In each case, we indicate the CPU time which measures the amount of work performed for one subdomain. The Wall Clock Time (WCT), a global measure which is more sensitive to external perturbations induced by the operating system and the presence of other users, was considered to be unreliable in many cases; so we mention it only for the first set of experiments. One should note that the gains calculated with WCT were always greater.

The CPU plots show the total time as well as the time dedicated to augmentation (preparation of the coarse operator, initialization and projections); the difference represents the iterations of the solver.

All the calculations used a dual formulation of the interface problem through domain decomposition (FETI). The convergence was evaluated using the norm of the residual (which corresponds to the displacement gaps at the interfaces) normalized by the condensed right-hand side. Classically for such structural problems, total reorthogonalization was used to enforce the $\boldsymbol{A}$-conjugation of the search directions. (The case without reorthogonalization is discussed briefly in the first example.)

5.1 The case of a sequence of linear systems

We considered a cube (of side 50 mm) with $4 \times 4 \times 4 = 64$ small cubic inclusions (of side 5.5 mm). A slice through this structure is shown in Figure 2. The cube was clamped over one side, and the opposite side plus another side were subjected to uniform pressure. The mesh consisted of $125,000$ linear hexahedral elements for a total of $400,000$ degrees of freedom. Three automatic decompositions (into 12, 48 and 96 subdomains) were performed using the Metis algorithm

[23] (see Figure 3). The resulting interface system contained 54,000 unknowns for the 12-subdomain decomposition, 96,000 unknowns for the 48-subdomain decomposition and 133,000 unknowns for the 96-subdomain decomposition. All the materials were isotropic, linear and elastic, and were characterized by their Young's modulus and Poisson's coefficient. The material properties of each inclusion and of the matrix were chosen randomly following a normal law with a relative standard deviation equal to 10%, leading to a $\pm 23\%$ variation range about the nominal value. The average Young's modulus was 200 MPa for the matrix and 20,000 MPa for each inclusion, and the average Poisson's coefficient was 0.27 for the matrix and 0.35 for each inclusion. The objective was to perform the calculations for 40 draws of the 130 coefficients.

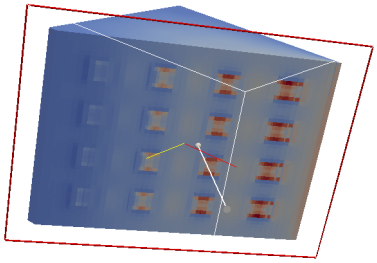

Figure 2: A slice through the heterogeneous cube (shear stress)

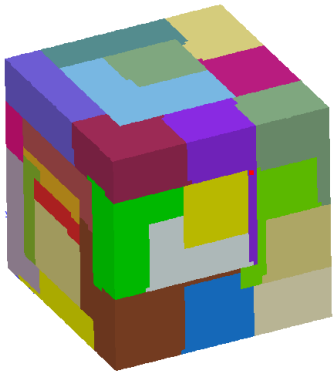

Figure 3: The decomposition into 48 subdomains

We used a dual formulation (FETI) with both a Dirichlet (optimal) and a lumped preconditioner, leading to 10^{-3} and 10^{-6} APCG accuracy respectively. We considered the following algorithms: conjugate gradients (cg), total reuse of Krylov subspaces (trks) and selective reuse of Krylov subspaces with two values of the criterion, $\varepsilon = 10^{-6}$ (srks6) and $\varepsilon = 10^{-14}$ (srks14). In addition, in the last case ($\varepsilon = 10^{-14}$), we also attempted to further refine the selection by not selecting the converged Ritz values contained in the central cluster (identified by the algorithm proposed by [32]); this method is labeled (clust14).

5.1.1 Comparison of the strategies

The results for the 12-subdomain decomposition are summarized in Table 1, which gives the average number of APCG iterations to convergence, the average size of the augmentation space, the final size of the augmentation space, the average CPU and wall clock times per system, from which we also deduced the augmentation time (operator preparation and projection). Computations were conducted one dual hexacore processor (one subdomain per core). When the average and final sizes of the augmentation space are close, this means that

			12 subdomains						
accur.	precond.		avg. # it	avg. n_c	max n_c	avg. total CPU	avg. CPU aug.	avg. total WCT	avg. WCT aug.
10^{-3}	Dirichlet	cg (no reo.)	70	—	—	23.5	0	31.8	0
		cg	44.3	—	—	16.0	0	24.5	0
		trks	2.4	77.8	96	7.1	6.3	9.1	7.6
		srks6	20.5	41.3	50	11.0	3.8	15.6	4.7
		srks14	25.6	24.6	27	**11.7**	2.7	**17.1**	3.5
		clust14	**30.6**	16.8	24	13.8	2.2	25.3	3.2
	Lumped	cg (no reo.)	145	—	—	27.7	0	40.3	0
		cg	68.1	—	—	13.6	0	24.0	0
		trks	**0.4**	71.8	74	**5.7**	5.6	**6.8**	6.7
		srks6	27.4	81	108	12.3	6.5	18.0	8.0
		srks14	32.0	59.6	71	11.8	5.1	18.1	6.4
		clust14	51.1	20	39	12.8	2.3	21.7	3.2
10^{-6}	Dirichlet	cg (no reo.)	174	—	—	58.3	0	85.8	0
		cg	84.4	—	—	30.0	0	49.1	0
		trks	13.2	382.9	551	46.5	41.3	60.4	52.8
		srks6	36.7	104.3	142	22.3	8.6	35.7	11.0
		srks14	42.8	72.6	87	22.7	6.3	36.2	8.1
		clust14	**60.8**	35.2	77	**25.7**	3.6	**39.8**	4.9
	Lumped	cg (no reo.)	>400	—	—	>78	0	>110	0
		cg	147.7	—	—	31.7	0	61.0	0
		trks	16.3	516.7	735	70.2	65.7	93.3	85.9
		srks6	54.2	225.7	311	33.4	20.5	49.4	25.6
		srks14	60.6	170.2	216	29.2	15.0	44.2	18.3
		clust14	**129.4**	20	39	**30.4**	2.6	**55.6**	4.0

Table 1: Performance summary for the cube with inclusions

most of the augmentation space was identified with the first systems. For a given configuration (accuracy and preconditioner), the figures in bold in the three columns 'average number of iterations', 'average CPU time' and 'average wall clock time' indicate the best strategy in terms of gain per unit augmentation vector compared to CG.

For the 12-subdomain decomposition, Figures (4, 6, 8) (for an objective of 10^{-3} accuracy) and Figures (5, 7, 9)) (for an objective of 10^{-6} accuracy) give the evolutions of the number of APCG iterations to convergence for each linear system, the dimension of the augmentation space n_c and the the CPU time for the resolution of each system, with both lumped and Dirichlet preconditioners.

Without full reorthogonalization, the performance was very poor and led to about twice the number of iterations of the recommended fully reorthogonalized

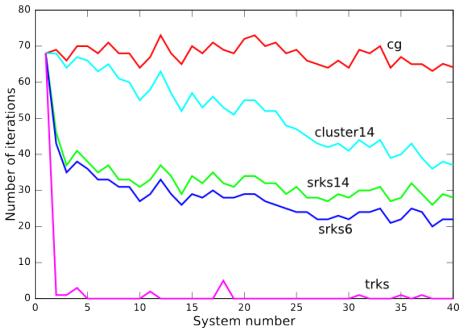

Figure 4: Cube 12 subdomains, lumped, $10^{-3}accuracy$, number of iterations per linear system

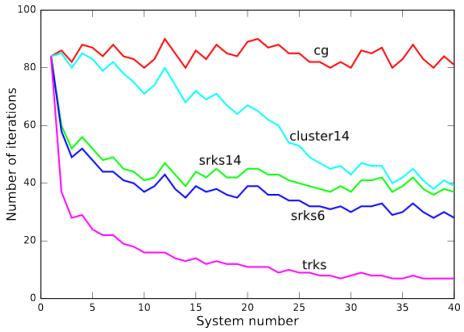

Figure 5: Cube 12 subdomains, Dirichlet, $10^{-6}accuracy$, number of iterations per linear system

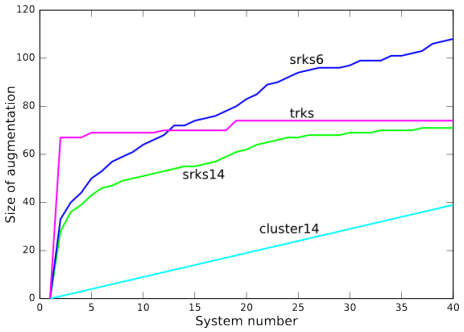

Figure 6: Cube 12 subdomains, lumped, $10^{-3}accuracy$, dimension of the augmentation space

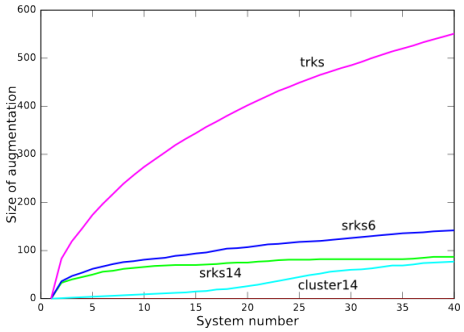

Figure 7: Cube 12 subdomains, Dirichlet, $10^{-6}accuracy$, dimension of the augmentation space

conjugate gradients. This was expected because systems resulting from domain decomposition formulations are known to often require full reorthogonalization [13]. Furthermore, one should note that the additional iterations carried out in the non-reorthogonalized case led to vector sets which made the Ritz analysis more complex due to the appearance of nonphysical, multiple eigenvalues. The non-reorthogonalized approach was no longer considered in the other examples.

With the TRKS approach, two types of behavior were observed. In the low-accuracy case (10^{-3}), for both preconditioners (but especially for the lumped preconditioner), the size of the augmentation space reached a plateau, which means that the augmentation space contained almost all the required information; the gains in terms of both the number of iterations ($> 90\%$) and the CPU time ($> 55\%$) were excellent. In the high-accuracy case (10^{-6}), the size of the augmentation space never stabilized; therefore, even though the number of iterations decreased drastically, the CPU time increased. Table 2 gives extended performance results for TRKS which confirm this analysis. The gains are given

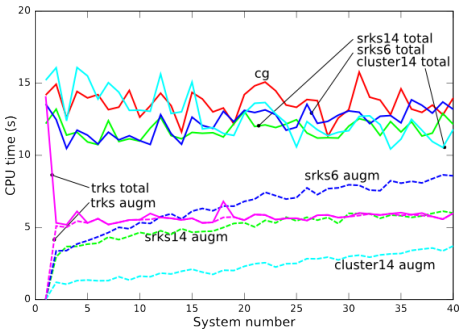

Figure 8: Cube 12 subdomains, lumped, $10^{-3}accuracy$, CPU time per linear system

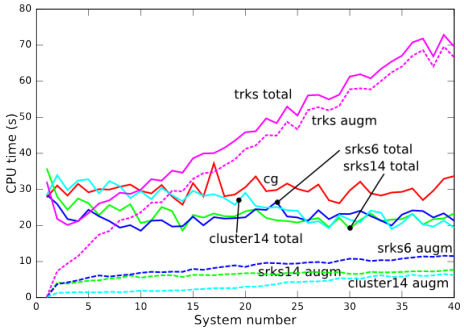

Figure 9: Cube 12 subdomains, Dirichlet, $10^{-6}accuracy$, CPU time per linear system

relative to conjugate gradients. The efficiency of augmentation is defined by the average decrease in the number of iterations per augmentation vector; the higher the required accuracy, the less efficient the TRKS approach. These results justify our decision to select the subspaces so that the dimension of the augmentation space would remain under control.

The SRKS14 approach succeeded in limiting the size of the augmentation space and led to a satisfactory decrease in the number of iterations. As can be seen on the figures, SRKS6 did not stabilize the augmentation space as efficiently and behaved half way between TRKS and SRKS14; therefore, we will choose SRKS14 as our reference algorithm from now on.

The cluster strategy as it stands today gave unsatisfactory results: even though it often led to the best gain per augmentation vector, it seemed to impair the selection of useful vectors and allow much less reduction in the number of iterations than SRKS. After the resolution of many systems, it tended to lead to the same augmentation space as SRKS.

To confirm that hypothesis, we compared the spaces C_{SRKS} and $C_{cluster}$ after the 40 resolutions for the low-accuracy Dirichlet case. We used the following procedure: first, the vectors were orthonormalized using SVD: $C = U\Sigma V^T$; then SVD was applied to the concatenated matrix $[U_{SRKS}, U_{cluster}]$. A plot of the singular values is shown in Figure 10. Independent spaces would lead to a constant value equal to 1, while for nested spaces the common space would lead to $\{\sqrt{2}, 0\}$ pairs of singular values. One can observe that the spaces are not exactly nested, but come quite close.

In conclusion, the cluster strategy is not mature yet, but it is promising. It was not considered for the following experiments because, due to the larger number of systems involved, it would behave quite similarly to SRKS.

5.1.2 Study of SRKS14 in various configurations

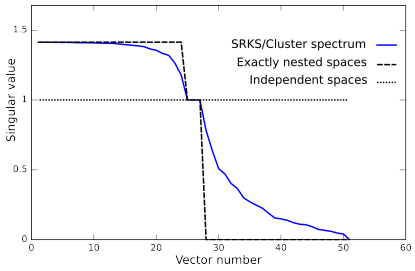

Figure 10: Singular values of $[C_{SRKS}, C_{cluster}]$

accur.	precond.	# subdomains	iteration gain	CPU gain	efficiency of augmentation
10^{-3}	lumped	12	99.4%	58.2%	0.94
		48	97.7%	60.1%	0.97
	Dirichlet	12	94.7%	55.5%	0.54
		48	96.2%	62.6%	0.69
10^{-6}	lumped	12	89%	-121.2%	0.25
		48	90.2%	-162.3%	0.37
	Dirichlet	12	84.4%	-55.1%	0.19
		48	83.3%	-90.7%	0.21

Table 2: Relative performance of TRKS

Table 3 shows the relative performance of SRKS14 as a function of the number of subdomains, of the preconditioners and of the accuracy. The efficiency of the augmentation is defined as the decrease in the average number of iterations per augmentation vector. One can observe that the efficiency ranged between 0.5 and 0.85 and was best for the lower accuracy and the improved preconditioner. For these spectra in which there exist no small isolated eigenvalues (which could lead to efficiencies greater than 1), such results are consistent with the theory (see Section 4.3). In the next section, we will see that this moderate efficiency does not preclude significant CPU improvements.

The gains in terms of the number of iterations were relatively stable, typically between 50% and 60% in the high-accuracy case.

5.1.3 Influence of the hardware configuration on the CPU gains

Now, let us study the performance of SRKS14 in terms of CPU time for the same decomposition into 48 subdomains, but using different hardware configurations:

1. Configuration A corresponds to 4 dual hexacore nodes with 1 subdomain

precond.	accur.	# subdomains	CG avg. # iterations	avg. n_c	iteration gain	efficiency of augmentation
lumped	10^{-3}	12	68.1	59.6	52.9%	0.6
		48	43.7	28.2	54.2%	0.84
	10^{-6}	12	147.7	170.2	59%	0.51
		48	162.7	186.3	62.5%	0.55
Dirichlet	10^{-3}	12	43.3	24.6	42.2%	0.76
		48	50.7	31	48.6%	0.79
		96	67.4	51.1	51.7%	0.68
	10^{-6}	12	84.4	72.6	49.3%	0.57
		48	116.	111.5	57.2%	0.6
		96	140.7	141.9	60.3%	0.6

Table 3: Iteration gains for SRKS14

per core;

2. Configuration B corresponds to 6 dual quadcore nodes with 1 subdomain per core;

3. Configuration C corresponds to 3 dual quadcore nodes with 2 subdomains per core;

4. Configuration D corresponds to 2 dual quadcore nodes with 3 subdomains per core.

One can note that the processors in Configuration A were different from those used in the other cases. In all the cases, the memory was sufficient to avoid swapping. The results are given in Table 4 for the Dirichlet preconditioner and in Table 5 for the lumped preconditioner. One can see that Configurations B,C and D had similar performances and were slower than Configuration A due to the different memory technology.

One interesting factor is the ratio of the average CPU cost of an iteration to the average CPU cost of an augmentation vector (the last columns of Table 4 and 5). One can see that in Configuration A, 4 augmentation vectors cost no more than one iteration; in the other configurations 7 augmentation vectors cost no more than one iteration. Since we saw that one needs about $1/0.6 \simeq 1.6$ augmentation vectors to save one iteration, the advantage of augmentation is clear. Indeed, we observe a 32% CPU improvement in Configuration A and a 40% to 50% improvement in the other configurations.

Note that when the lumped preconditioner is used the equivalent cost of an iteration is only 2.8 augmentation vectors in Configuration A and 4.5 augmentation vectors in Configuration D (see Table 5). Since the efficiency of the augmentation vectors is less when this inexpensive preconditioner is used (in the high accuracy case), so is the CPU improvement.

Configuration	CG avg. CPU	CPU gain	CPU per iteration / CPU per augm. vector
A	9.5	32.6%	4
B	25.7	41.5%	6.7
C	29.7	48.9%	7.7
D	29.9	47.6%	7.8

Table 4: CPU performance of SRKS14 for 10^{-6} accuracy with the Dirichlet preconditioner

Configuration	CG avg. CPU	CPU gain	CPU per iteration / CPU per augm. vector
A	10.7	22.4%	2.8
D	27.4	33.3%	4.5

Table 5: CPU performance of SRKS14 for 10^{-6} accuracy with the lumped preconditioner

The ratio of the CPU time per iteration to the CPU time per augmentation vector for SRKS (Column 4 of the previous tables) turned out to be relatively stable for a given machine with a given preconditioner. This is due to the stability of the size of the augmentation space which prevented the cost from soaring (as would happen with TRKS). Thus, the CPU performance can be deduced from the iteration gains and the augmentation efficiency (see Table 3). For instance, the CPU gain for SRKS with the 96-subdomain decomposition was slightly greater than 50%.

5.2 The case of a sequence of nonlinear problems

Now let us consider a hexahedral holed plate ($10\times10\times0.2$ mm with a center hole of radius 1 mm, see Figure 11) subjected to unidirectional tension (a prescribed normal displacement). The plate was discretized into $61,000$ linear hexahedral elements for a total of $41,000$ degrees of freedom. The structure was divided into 8 subdomains using the Metis algorithm, which resulted in an interface system with $3,000$ unknowns. The problem was solved using one 8-core processor (one subdomain per core). Elastic-plastic behavior with nonlinear isotropic hardening and a Von Mises'-type plasticity criterion was assumed. Denoting σ the Cauchy stress tensor, $\epsilon(u)$ the symmetric gradient of the displacement field u, and $\mathcal{K}$ the Hooke tensor, the material law can be written as:

$$\left\{\begin{array}{l} \epsilon(u) = \epsilon^e + \epsilon^p, \qquad \sigma = \mathcal{K} : \epsilon^e \\ \text{if } f(\sigma) = 0 \text{ then } \dot{\epsilon}^p = \lambda f_{,\sigma} \\ \text{if } f(\sigma) \leqslant 0 \text{ then } \dot{\epsilon}^p = 0 \\ f(\sigma) = \sqrt{\frac{3}{2}\,\sigma : \sigma} - \left(R_0 + Q\left(1 - e^{-b\lambda}\right)\right) \end{array}\right. \tag{23}$$

The coefficients were assigned a normal law with a 10% relative standard deviation, which implied variations of up to ±23% in the coefficients. The mean values of the material parameters were: $E = 200,000$ MPa, $\nu = 0.3$, $R_0 = 300$ MPa, $b = 22$ and $Q = 170$ MPa. The loading was applied in two steps: first, a single increment to reach the elastic limit; then, 16 equal increments in order to multiply the prescribed displacement by 4. The objective of the study was to analyze 21 configurations.

Again, the linear solver used was FETI with a Dirichlet or lumped preconditioner. The accuracy objective for the linear systems was set at 10^{-6}. (The accuracy must be high for the nonlinear process to run well). Because of the approximations, not all the methods converged in the same number of Newton iterations; on average, one nonlinear analysis required the resolution of 95 tangent systems. Table 6 summarizes the performances of the various methods; Figure 12 shows the evolution of the average number of APCG iterations with the lumped preconditioner during the sequence of linear systems; Figure 13 shows the evolution of the size of the augmentation space; Figures 14 and 15 show the evolutions of the average CPU time and wall clock time for the resolution of one linear system along with the evolution of the average augmentation time (operator creation and projection).

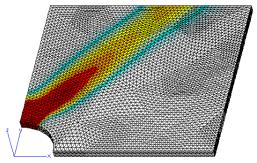

Figure 11: The holed plate example (plastic strain)

precond.	method	avg. # it	avg. n_c	max n_c	avg. CPU	avg. WCT
Dirichlet	cg	25.6	–	–	1.21	3.03
Dirichlet	trks*	1.4	358	492	2.66	9.83
Dirichlet	srks14	**16.1**	17	19	**0.98**	**2.35**
lumped	cg	41.4	–	–	1.03	3.24
lumped	trks*	1.2	520	695	4.72	6.98
lumped	srks14	**19.1**	43	45	**0.87**	**2.08**

* calculation too slow, was stopped before all the systems were solved

Table 6: Holed plate, performance summary

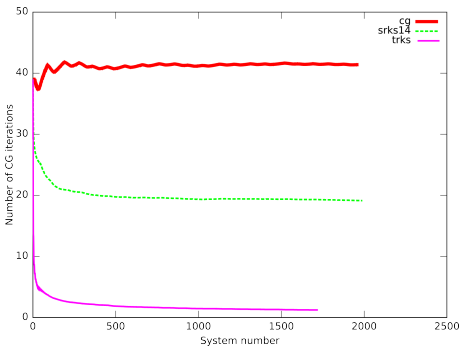

Figure 12: Plate, lumped – avg. # it. / linear system

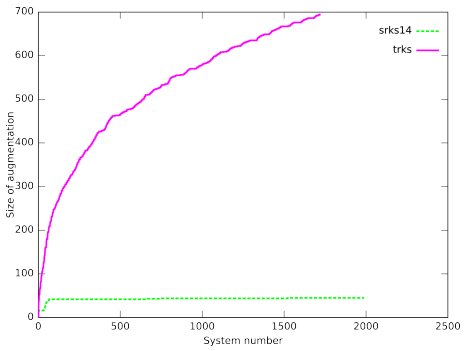

Figure 13: Plate, lumped – dimension of aug. space

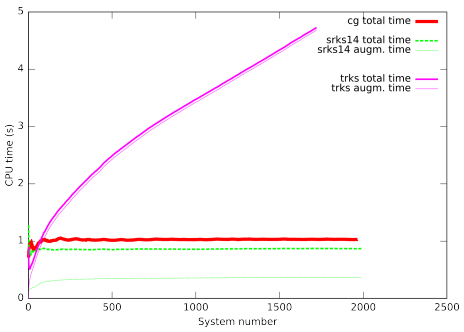

Figure 14: Plate, lumped – avg. CPU time / system

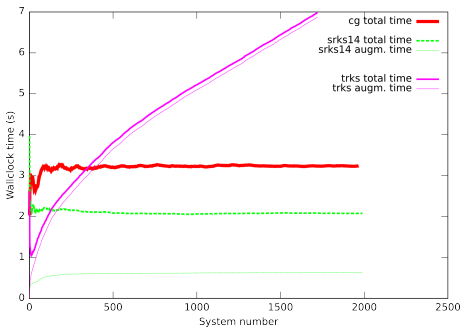

Figure 15: Plate, lumped – avg. wall clock time / system

This test leads to conclusions similar to the previous ones. More specifically, one can observe that the SRKS augmentation space selected after the first nonlinear configuration remained stable. Conversely, since the TRKS augmentation space never reached a plateau, the solutions of the linear systems did not belong to a common space. SRKS was the most efficient method, leading to a 20% CPU gain and a 36% wall clock time improvement.

5.3 The case of a large displacement problem

Finally, let us consider the problem of the buckling of a straight heterogeneous beam with a circular cross section (length/diameter ratio equal to 30), clamped at one end and subjected to an axial pressure at the other, with no radial displacement. The heterogeneities consisted of five straight fibers whose stiffness was $1,000$ times that of the matrix. The problem was formulated in the updated Lagrangian framework, assuming linear elastic behavior (characterized by the Young's modulus and Poisson's coefficient) in the current configuration. The beam was discretized into $90,000$ linear hexahedral finite elements for a total of $300,000$ degrees of freedom. It was divided into 10 subdomains using the Metis algorithm, leading to an interface system with $16,000$ unknowns. A single 12-core processor was used (1 subdomain per core, leaving 2 inactive cores). The pressure was applied incrementally up to the configuration shown in Figure 18, in which the maximum axial displacement was about 3% of the total length. 12 increments were used, leading to the resolution of about 30 tangent linear systems.

We used a FETI solver with a Dirichlet preconditioner and an "identity" projector. The FETI convergence criterion was set to 10^{-6}. Figure 16 shows the evolution of the number of conjugate gradient iterations required for the resolution of each linear system. Figure 17 shows the evolution of the size of the augmentation space. Three algorithms were tested: classical conjugate gradients, total reuse of subspaces, and selective reuse of subspaces ($\varepsilon = 10^{-14}$). Table 7 summarizes the main results.

The following observations can be made:

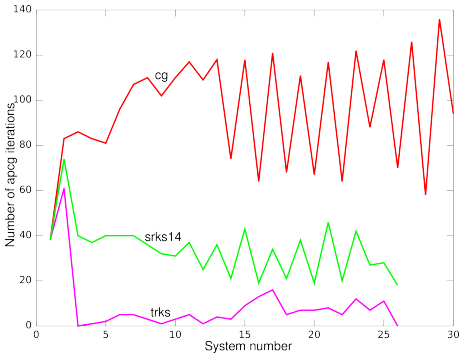

Figure 16: Buckling of the heterogeneous beam: number of iterations per linear system

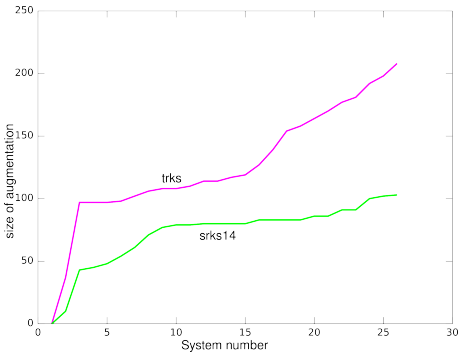

Figure 17: Buckling of the heterogeneous beam: dimension of the augmentation space for each linear system

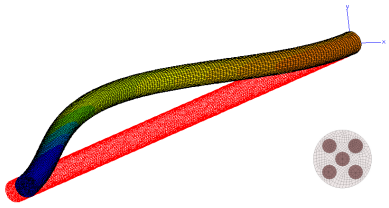

Figure 18: The beam in the reference and deformed configurations, with a view of the cross section

- The performance of TRKS was really impressive because the systems converged in 10 times fewer iterations than with CG, but the size of the associated augmentation space was large (up to 208 vectors) and never ceased to increase;
- SRKS appeared to be efficient: the number of iterations was divided by 3 with a space whose size increased slowly, then reached a plateau;
- With the hardware configuration used, the best results in terms of computation time (a 48% CPU improvement) were achieved with TRKS, but the gain normalized by the number of augmentation vectors was better with SRKS (a 29% CPU improvement). SRKS was truly successful in controlling the dimension of the augmentation space.

In this example, the augmentation proved to be very efficient in terms of the iteration gain per augmentation vector, especially for SRKS (0.85). This was probably because of a specificity of the spectrum of the preconditioned operator due to the use of domain decomposition in large displacements. Indeed, the tangent matrix of a floating subdomain with prescribed Neumann conditions

method	avg. # it.	avg. n_c	CPU	WCT
CG	97.2	–	392	504
TRKS	7.8	131.7	203	217
SRKS 14	**33.8**	75.1	**278**	**298**

Table 7: Recycling performance for the buckling problem

may become non-positive, contrary to the Dirichlet operator which remains positive definite. It is known that a slight lack of positivity of the operator does not prevent reorthogonalized conjugate gradients from converging [37], but convergence is slower than when all the eigenvalues are positive. The positivity of the preconditioner makes the selection procedure still possible. Moreover, the negative eigenvalues are systematically selected by the procedure, so using augmentation causes the solver to iterate in the subspace in which the operator is positive, leading to a much better convergence rate.

6 Conclusion

This paper dealt with the resolution of sequences of large linear systems with varying matrices and right-hand sides using conjugate gradients. We proposed several algorithms based on the augmentation of the current Krylov subspace by a selection of previously generated subspaces. The advantage of these methods is that some of the iterations are replaced by the preprocessing of a coarse problem associated with optimized operations.

When low accuracy is sufficient, total reuse of the previous subspaces (the TRKS algorithm) appears to lead to satisfactory results. When high accuracy is required, the subspaces are too unstable, which causes the dimension of the TRKS augmentation space to soar. Therefore, we proposed to retain only the part of the subspace generated by the Ritz vectors associated with converged Ritz values of the preconditioned operator (the SRKS algorithm). These vectors can be built very inexpensively. Such an augmentation was found to remain stable throughout the linear systems and to lead to a reduction in the number of iterations which is consistent with the theory. In terms of computation time, the proposed method leads to a variable, but always positive, gain compared to non-augmented systems. We observed CPU time improvements of 20% to 50%, and wall clock time improvements of 40% to 70%.

Up until now, our attempts to improve the selection algorithm by eliminating the converged values in the central part of the spectrum have not led to impressive results. This probably means that the Ritz vectors associated with converged Ritz values contain meaningful information which cannot be removed from the analysis of the current system. A continuation of this work could consist in a better analysis of the accumulation of the augmentation vectors. This can be done by studying the coarse matrix $\boldsymbol{C}^T\boldsymbol{A}\boldsymbol{C}$ whose distance to the identity matrix characterizes the variation of the Krylov subspaces. Another objective

would be to port some of the ideas presented in this paper to nonsymmetric solvers.

References

[1] J. I. Aliaga, D. L. Boley, R. W. Freund, and V. Hernández. A Lanczos-type method for multiple starting vectors. *Mathematics of Computation*, 69:1577–1601, 2000.

[2] O. Axelsson and G. Lindskog. On the rate of convergence of the preconditioned conjugate gradient method. *Numerische Mathematik*, 48:499–523, 1986.

[3] A. H. Baker, J. M. Dennis, and E. R. Jessup. On improving linear solver performance: A block variant of GMRes. *SIAM Journal on Scientific Computing*, 27(5):1608–1626, 2006.

[4] J.F. Carpraux, S. Godunov, and S. Kuznetsov. Stability of the Krylov bases and subspaces. Technical Report 2296, INRIA, 1994.

[5] A. Chapman and Y. Saad. Deflated and augmented Krylov subspace techniques. *Numerical Linear Algebra with Applications*, 4(1):43–66, 1997.

[6] Z. Dostal. Conjugate gradient method with preconditioning by projector. *International Journal of Computer Mathematics*, 23:315–323, 1988.

[7] J. Erhel, K. Burrage, and B. Pohl. Restarted GMRes preconditioned by deflation. *Journal of Computational and Applied Mathematics*, 69:303–318, 1996.

[8] J. Erhel and F. Guyomarc'h. An augmented conjugate gradient method for solving consecutive symmetric positive definite linear systems. *SIAM Journal on Matrix Analysis and Applications*, 21(4):1279–1299, 2000.

[9] C. Farhat, P.-S. Chen, and F.-X. Roux. The two-level FETI method - part II: Extension to shell problems. parallel implementation and performance results. *Computer Methods in Applied Mechanics and Engineering*, 155:153–180, 1998.

[10] C. Farhat, L. Crivelli, and F. X. Roux. Extending substructure based iterative solvers to multiple load and repeated analyses. *Computer Methods in Applied Mechanics and Engineering*, 117:195–209, 1994.

[11] C. Farhat and J. Mandel. The two-level FETI method for static and dynamic plate problems - part I: An optimal iterative solver for biharmonic systems. *Computer Methods in Applied Mechanics and Engineering*, 155:129–152, 1998.

[12] C. Farhat, K. Pierson, and M. Lesoinne. The second generation FETI methods and their application to the parallel solution of large-scale linear and geometrically non-linear structural analysis problems. *Computer Methods in Applied Mechanics and Engineering*, 184(2-4):333–374, 2000.

[13] C. Farhat and F. X. Roux. Implicit parallel processing in structural mechanics. *Computational Mechanics Advances*, 2(1):1–124, 1994. North-Holland.

[14] R. Freund. Model reduction methods based on Krylov subspaces. *Acta Numerica*, 12:267–319, 2003.

[15] Roland Freund. Krylov-subspace methods for reduced-order modeling in circuit simulation. *Journal of Computational and Applied Mathematics*, 123(1-2):395 – 421, 2000.

[16] P. Gosselet and C. Rey. On a selective reuse of Krylov subspaces in Newton-Krylov approaches for nonlinear elasticity. In *Proceedings of the* 14^{th} *conference on domain decomposition methods*, pages 419–426, Cocoyoc, Mexico, 2002.

[17] P. Gosselet and C. Rey. Non-overlapping domain decomposition methods in structural mechanics. *Archives of computational methods in engineering*, 13(4):515–572, 2007.

[18] P. Gosselet, C. Rey, P. Dasset, and F. Léné. A domain decomposition method for quasi incompressible formulations with discontinuous pressure field. *Revue européenne des élements finis*, 11:363–377, 2002.

[19] P. Gosselet, C. Rey, and D. Rixen. On the initial estimate of interface forces in FETI methods. *Computer Methods in Applied Mechanics and Engineering*, 192:2749–2764, 2003.

[20] P. Gosselet, D. Rixen, and C. Rey. A domain decomposition strategy to efficiently solve structures containing repeated patterns. *International Journal for Numerical Methods in Engineering*, 78(7):828–842, 2009.

[21] P.J. Heres, D. Deschrijver, W.H.A. Schilders, and T. Dhaene. Combining Krylov subspace methods and identification-based methods for model order reduction. *International Journal of Numerical Modelling : Electronic Networks, Devices and Fields*, 20(6):271–282, 2007.

[22] Z. Jia. The convergence of harmonic Ritz values, harmonic Ritz vectors and refined harmonic Ritz vectors. *Mathematics of computation.*, 74(251):1441–1456, 2004.

[23] G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing*, 20(1):359–392, 1998.

[24] M. E. Kilmer and E. de Sturler. Recycling subspace information for diffuse optical tomography. *SIAM Journal on Scientific Computing*, 27(6):2140–2166, 2006.

[25] A. Klawonn and O.B. Widlund. FETI and Neumann-Neumann iterative substructuring methods: connections and new results. *Communications on Pure and Applied Mathematics*, LIV:0057–0090, 2001.

[26] P. Le Tallec, J. Mandel, and M. Vidrascu. A Neumann-Neumann domain decomposition algorithm for solving plate and shell problems. *SIAM Journal on Numerical Analysis*, 35(2):836–867, April 1998.

[27] F. Léné and C. Rey. Some strategies to compute elastomeric lamified composite structures. *Composite structures*, 54:231–241, 2001.

[28] F. J. Lingen. Efficient Gram-Schmidt orthonormalisation on parallel computers. *Communication in Numerical Methods in Engineering*, 16:57–66, 2000.

[29] J. Mandel. Balancing domain decomposition. *Communication in Numerical Methods in Engineering*, 9:233–241, 1993.

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

[31] M. Meyer and H. G. Matthies. Efficient model reduction in non-linear dynamics using the karhunen-loeve expansion and dual-weighted-residual methods. *Computational Mechanics*, 31(1-2):179–191, 2003.

[32] N. Molinari, C. Bonaldi, and J.P. Daurès. Multiple temporal cluster detection. *Biometrics*, 57:577–583, 2001.

[33] R. B. Morgan. A restarted GMRes method augmented with eigenvectors. *SIAM Journal on Matrix Analysis and Applications*, 16:1154–1171, 1995.

[34] Northwest Numerics. *Z-set user manual*, 2001.

[35] Y. Notay. On the convergence rate of the conjugate gradients in presence of rounding errors. *Numerische Mathematik*, 65:301–317, 1993.

[36] A. Nouy. A generalized spectral decomposition technique to solve a class of linear stochastic partial differential equations. *Computer Methods in Applied Mechanics and Engineering*, 196(45-48):4521–4537, 2007.

[37] C. Paige. Approximate solutions and eigenvalue bounds from Krylov subspaces. *Numerical Linear Algebra with Applications*, 2(2):115–133, 1995.

[38] M. Parks, E. De Sturler, G. Mackey, D. Johnson, and S. Maiti. Recycling Krylov subspaces for sequences of linear systems. *SIAM Journal on Scientific Computing*, 28(5):1651–1674, 2006.

[39] C. Rey and F. Risler. A Rayleigh-Ritz preconditioner for the iterative solution to large scale nonlinear problems. *Numerical Algorithms*, 17:279–311, 1998.

[40] Christian Rey. An acceleration technique for the solution of non-linear elasticity problems by domain decomposition. *Comptes Rendus de l'Académie des Sciences, Paris, série IIb*, 322(8):601–606, 1996.

[41] Franck Risler and Christian Rey. Iterative accelerating algorithms with Krylov subspaces for the solution to large-scale nonlinear problems. *Numerical Algorithms*, 23:1–30, 2000.

[42] D. Ryckelynck, F. Chinesta, E. Cueto, and A. Ammar. On the "a priori" model reduction: Overview and recent developments. *Archives of Computational Methods in Engineering*, 13:91–128, 2006.

[43] Y. Saad. Analysis of augmented Krylov subspace methods. *SIAM Journal on Matrix Analysis and Applications*, 18(2):435–449, April 1997.

[44] Y. Saad. *Iterative methods for sparse linear systems*. SIAM, Philadelphia, USA, 2nd edition, 2003.

[45] Y. Saad. *Numerical Methods for Large Eigenvalue Problems*, volume 66 of *Classics in Applied Mathematics*. SIAM, Philadelphia, USA, revised edition, 2011.

[46] Y. Saad, M. Yeung, J. Erhel, and F. Guyomarc'h. A deflated version of the conjugate gradient algorithm. *SIAM Journal on Scientific Computing*, 21(5):1909–1926, 2000.

[47] V. Simoncini and D. Szyld. Recent computational developments in Krylov subspace methods for linear systems. *Numerical Linear Algebra with Applications*, 14(1):1–59, 2007.

[48] J.M. Tang, R. Nabben, C. Vuik, and Y.A. Erlangga. Theoretical and numerical comparison of various projection methods derived from deflation, domain decomposition and multigrid methods. Reports of the Department of Applied Mathematical Analysis 07-04, Delft university of technology, 2007.

[49] A. van der Sluis and H. van der Vorst. The rate of convergence of conjugate gradients. *Numerische Mathematik*, 48:543–560, 1986.

[50] S. Wang, E. De Sturler, and G. Paulino. Large-scale topology optimization using preconditioner Krylov subspace methods with recycling. *International Journal for Numerical Methods in Engineering*, 69(12):2441–2468, 2007.

[51] Jia Z. and G.W. Stewart. On the convergence of the Ritz values, Ritz vectors and refined Ritz vectors. Technical Report 3896, Institute of Advanced Computer Studies, Department of Computer Science, University of Maryland at College Park, 1999.