Application of a recycling Krylov subspace strategy for discrete element method applied to brittle crack problems

Sylvain Gavoille — Arnaud Delaplace — Christian Rey

LMT-Cachan (ENS Cachan/CNRS/Université Paris 6/UniverSud Paris)
61, avenue du Président Wilson, F-94235 Cachan
{delaplace, rey, gavoille}@lmt.ens-cachan.fr

ABSTRACT. *This paper deals with a comparative study of two iterative Krylov solvers (GIRKS and SRKS) dedicated to the solution to a sequence of large linear problems. We apply these two algorithms to brittle crack problems modelized with a discrete element method. We show that these algorithms still reduce the total number of iterations but not the total CPU time. By considering the specific modification of the stiffness matrix for discrete modeling, we propose a simple evolution of the SRK algorithm leading to a reduction of the factor time (greater than* 2*). Efficiency of the algorithm is illustrated on 2D and 3D examples of crack propagation.*

RÉSUMÉ. *Ce papier propose une étude comparative de deux solveurs itératifs (GIRKS et SRKS) de type Krylov dédiés à la résolution d'une succession de grands systèmes linéaires. Nous appliquons ces algorithmes à des problèmes de rupture fragile traités avec un modèle aux éléments discrets. Nous montrons que hors du cadre des approches par décomposition de domaine, ces algorithmes restent performants en termes de réduction du nombre d'itérations, mais plus en termes de temps CPU. En considérant les modifications de la matrice de raideur propre au modèle discret, nous proposons une évolution simple de l'algorithme SRKS permettant de retrouver une réduction du temps de calcul par au minimum un facteur* 2*. Les performances de l'algorithme sont illustrées sur des exemples 2D et 3D de fissuration.*

KEYWORDS: *discrete element model, iterative solver, recycling Krylov subspaces.*

MOTS-CLÉS : *modèle discret, solveur itératif, sous-espaces de Krylov.*

DOI:10.3166/EJCM.18.647-667

1. Introduction

Discrete element method (DEM) has been coming back into favor during these last ten years when dealing with brittle heterogeneous material behavior (see for example (Van Mier *et al.*, 2002; Potyondy *et al.*, 2004; Yip *et al.*, 2006; Delaplace *et al.*, 2007)). Although the method has been introduced three decades ago (Cundall *et al.*, 1979), nowadays computer performance allows computation of structure behavior at a representative scale. One of the advantage of the approach is to take naturally into account material heterogeneity, by representing randomness of the miscrostructure. Another one is to represent material like an assembly of particles rather than a continuous media, allowing realistic description of discontinuity such as a crack.

A drawback of this material description at fine scale is that a large number of elements should be considered when performing simulations, leading to a large size stiffness matrix. This point is not a real problem when dealing with high rate loading, for which discrete element methods have a real interest to describe phenomena like fragmentation or spalling : in this case, explicit time integration schemes are usually preferred and do not required matrix inversion. On the other hand, when dealing with low rate dynamic problems for which implicit schemes are used, a large size system resolution should be performed for each time step and computational time can become discouraging. It is the same case for static problems where many successive system resolutions are performed during loading. Different algorithms have been proposed to address this problem. The first pragmatic method is to make a full inversion for the first resolution, and to use for example a Shermann-Morrison formula to compute inverse matrix for next resolutions. This method has been successfully used for large system size, but only for 2D problems (Nukala *et al.*, 2005). 3D problems need a too large amount of computer storage to be achieved. Then, all other methods are based on iterative solvers. Note that if periodic boundary conditions are considered, high efficiency algorithms have been proposed (Nukala *et al.*, 2004).

In this paper, we explore the possibility offered by the use of Krylov iterative solvers and more precisely by two different recycling Krylov subspace strategies : the generalized iterative reuse of Krylov subspace (GIRKS) algorithm ((Rey, 1996; Risler *et al.*, 2000)) and the selective reuse of Krylov subspace (SRKS) approach ((Rey *et al.*, 1998; Gosselet *et al.*, 2002)). These strategies are based on the reuse (or recycling) of numerical information coming from the solution to previous linear systems. Dedicated to the solution to a sequence of linear system, they have been able to accelerate significantly the solution to a sequence of large linear problems. Nonetheless, theses approaches have been introduced and validated in the context of domain decomposition method (see (Farhat *et al.*, 1994; Gosselet *et al.*, 2006) among other) which operates on a reduced-size problem. Beside they have been associated to a full re-orthogonalization procedure (Gram-Schmid process) which may not be required in other context. Hence, we first analyse the efficiency both in term of iteration number (in our study, the iteration number means the number of iterations required to reach a given precision) and CPU time of such recycling strategy without the use of domain decomposition context. Besides, considering specific changes in the operators for the

discrete element approach, a simple modified algorithm based on the SRKS approach and leading to an efficient recycling strategy is also introduced.

In the first part of the paper, we introduce the discrete element model used for this study. The elastic prediction algorithm used to solve static problem is proposed. In a second part, we present three generic problems, a small-size 2D tension test, a large-size 2D three-point bend test and a large-size 3D tension test, used for benchmarking the solver. In a third part, the reference iterative solver and the two recycling strategies are introduced. Their efficiency is analyzed on a first example. A modified algorithm based on the selective reuse of Krylov subspaces approach is then introduced. In a final part, the efficiency of the modified algorithm is illustrated with two large-scale examples.

2. Discrete model

2.1. *Basic principle*

We present in this part the model, based on a representation of the material as a particle assembly. Voronoi polygons (or polyhedra in 3D) are used, which allow fast mesh generation : (i) the particle nuclei are randomly generated in each cell of a regular grid, (ii) a Voronoi tesselation is performed using these nuclei. This grid support algorithm, proposed by (Moukarzel *et al.*, 1992), simplifies applications of boundary and loading conditions. Particles are considered rigid but they can overlap to reproduce material deformation. Cohesive forces are represented by beams joining the centers of each couple of neighbouring particles. Material elastic constants E and ν, respectively the Young's modulus and the Poisson coefficient, are related to beams elastic modulus and quadratic inertia factor. Elastic modulus is chosen to be the same for all beams. In the following, Euler-Bernoulli beams are considered, which are strictly equivalent to particle interface forces within the framework of elasticity and small strains.

Material nonlinear behavior is obtained by considering a perfect brittle behavior for beams (Figure 1). This choice is justified when considering damage behavior of the studied material. The breaking criterion for beam ij, that relies particle i and particle j, is written (Herrmann *et al.*, 1990; D'Addetta, 2004) :

$$P_{ij} = \left(\frac{\varepsilon_{ij}}{\varepsilon_{ij}^{cr}}\right)^2 + \left(\frac{|\theta_i - \theta_j|}{\theta_{ij}^{cr}}\right) \geq 1 \qquad [1]$$

ε_{ij} is the beam longitudinal strain, θ_i and θ_j are respectively rotations of particle i and particle j. ε_{ij}^{cr} and θ_{ij}^{cr} are two model parameters that control the material asymmetric behavior in tension and compression (Delaplace, 2009).

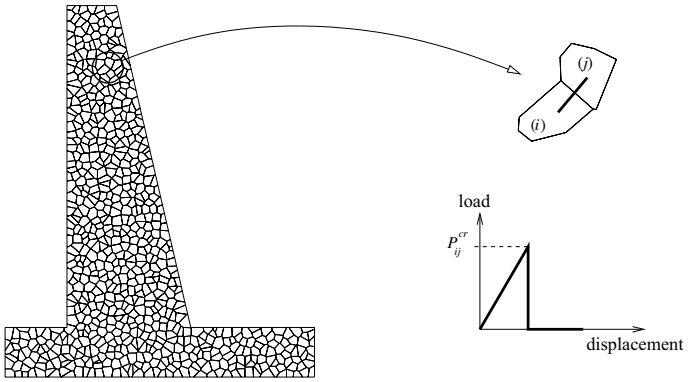

Figure 1. *A mesh sample with the representation of the brittle behavior*

2.2. *Algorithms for static problems*

As mentionned in introduction, our study deals with static problems, *i.e.* that inertia forces could be neglected. For the loading step k, the formulation of the problem is reduced to the equation :

$$\mathbf{K}^k\mathbf{u}^k = \mathbf{f}^k \qquad [2]$$

where $\mathbf{u}^k$ is the vector containing the particle degrees of freedom (dof) and $\mathbf{f}^k$ is the loading vector. $\mathbf{K}^k$ is the global stiffness matrix that depends on the loading step. A major property of this matrix is that evolution of different components is not continuous. For the chosen elastic brittle behavior, local stiffness matrix of each bond is either constant (surviving bond) or null (broken bond).

Equation [2] could be solved by two algorithms when $\mathbf{f}^k$ evolves, the monotonic algorithm or the elastic prediction one.

2.2.1. *Monotonic algorithm*

This is probably the most employed algorithm. Response is computed from a succession of increasing loading steps $\mathbf{f}^k$, $k \in \{1, ..., n_{\text{rupt}}\}$ with n_{rupt} the total number of loading steps before rupture. During each increment, a number m_k of beams can fail if their breaking threshold is exceeded. With this algorithm, the response depends on the incremental step. The algorithm for the step k is the following :

Step k

1) apply loading $\mathbf{f}^k$

2) compute $\mathbf{u}^k$ solving equation [2]

3) save couple $(\mathbf{u}^k, \mathbf{f}^k)$

4) find the m_k links that verify

$$P_{i_p j_p} \geq 1 \qquad p \in \{1, .., m_k\}$$

5) change stiffness matrix with

$$\mathbf{K}^{k+1} = \mathbf{K}^k - \sum_{p=1}^{m_k} \mathbf{L_{i_p j_p}}^T \mathbf{K^k_{i_p j_p}} \mathbf{L_{i_p j_p}}$$

where $\mathbf{L_{i_p j_p}}$ is the connectivity matrix of element $i_p j_p$.

2.2.2. *Elastic prediction algorithm*

This algorithm could be seen as an event-driven algorithm. Loading is not controlled by an increasing quantity (*e.g.* force or displacement) but by the decreasing of the secant stiffness. Usually, one link is broken for one step. This kind of control allows to follow snap back response, and uniqueness of solution is guaranteed in the sense that it does not depend of a loading parameter. By considering the elastic brittle law for bonds, we know exactly the number of changing elements in global stiffness matrix : $4 \times n_{\text{dof}}^2$ where n_{dof} is the number of degrees of freedom for one particle.

The algorithm for breaking the kth-bond is the following one :

Step k

1) apply elastic loading $\mathbf{f}^{el}$

2) compute $\mathbf{u}^{el}$ solving equation [2]

3) compute $\alpha_{\min}$ with

$$\alpha_{\min} = \min_{\substack{i,j \in \{1,..,n\} \\ i<j}} \left(\frac{1}{P_{ij}} \right)$$

4) save couple $(\alpha_{\min} \mathbf{u}^{el}, \alpha_{\min} \mathbf{f}^{el})$

5) change stiffness matrix with

$$\mathbf{K}^{k+1} = \mathbf{K}^k - \mathbf{L_{ij}}^T \mathbf{K^k_{ij}} \mathbf{L_{ij}}$$

where $\mathbf{L_{ij}}$ is the connectivity matrix of the particular element ij that satisfies step 3).

Responses obtained with these two algorithms are plotted on Figure 2. If a sufficient number of steps is considered with the monotic loading algorithm, both responses are similar. Note that just the elastic-prediction algorithm is able to follow the local instability (snapback) during the loading. Furthermore, nonphysical crack path could be obtained if the monotonic algorithm is used with a too large loading step, as shown on Figure 3 where two cracks are obtained. Because of its better accuracy to deal with elastic brittle behaviour, we will use the elastic-prediction algorithm for solving next problems. Then, solution to a problem is obtained by solving n_{rupt}-times equation [2], with n_{rupt} the total number of breaking bonds at global failure.

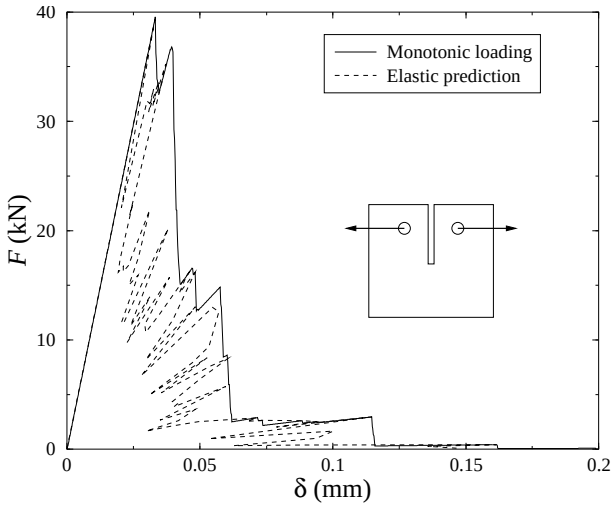

Figure 2. *Force-displacement response obtained with the two algorithms for a DCB test on a notched specimen*

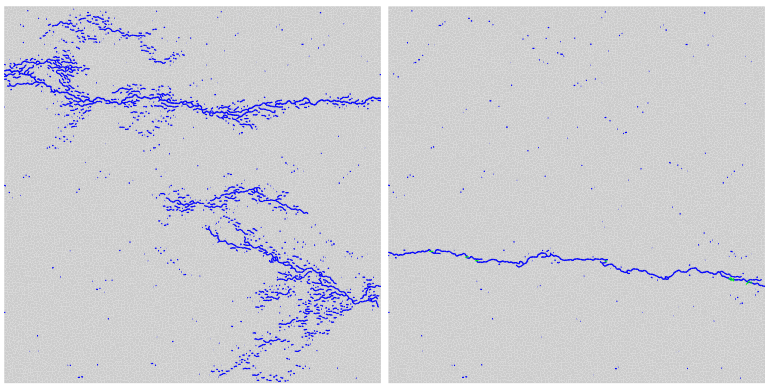

Figure 3. *Unphysical crack path (left) obtained with too large loading step monotonic algorithm, compared to the right crack path obtained with elastic prediction algorithm*

2.3. *Reference problems*

In order to illustrate easily the solver features, we first introduce a simple problem, called P0, with a limited size. Then, we propose two problems, PA and PB, with a more realistic number of degrees of freedom to illustrate the performance of the improved

solver. The number of degrees of freedom of the two problems are close, but the first one is solved in 2D as the second one is solved in 3D. The main initial properties of the stiffness matrices are given in Table 1.

2.3.1. *Tension test on a square (problem 0)*

This small-size problem is a simple 2D tension test on a square, with 64×64 particles, that corresponds to $12\,288$ degrees of freedom. After a first stage of distributed damage, a macrocrack appears and propagates through the sample. The number of systems to solve n_{rupt} can be estimated by the expression $64 \times \bar{n}_{\text{c}}^{2D}/2$, where $\bar{n}_{\text{c}}^{2D} \approx 6$ is the average number of connections per particle for a Voronoi tesselation in 2D.

2.3.2. *3-point bend test (problem A)*

Figure 4 shows the geometry and loading conditions of this test. The simulation is performed with a 2D-mesh of 800×100 particles, that corresponds to $240\,000$ degrees of freedom. The stiffness matrix has a narrow band diagonal form. In this problem, damage localizes from the beginning of the load, with a crack that propagates from the bottom of the beam. Total failure is obtained with a short number of broken fibers ($n_{\text{rupt}} \approx 100 \times \bar{n}_{\text{c}}^{2D}/2$).

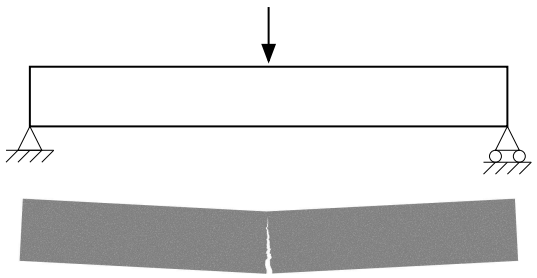

Figure 4. *Geometry and loading conditions for the 3-point bend test (PA), and crack pattern at the end of the loading*

2.3.3. *Tension test on a cube (problem B)*

Figure 5 shows the geometry and loading conditions of this test : a displacement is applied on the top face of a cube. Like for P0, after a first stage of distributed damage, a macrocrack appears through the sample, perpendicular to the loading direction. The simulation is performed with a 3D-mesh of $40 \times 40 \times 40 = 64\,000$ particles, that corresponds to $384\,000$ degrees of freedom. Total failure is obtained with a large number of broken fibers ($n_{\text{rupt}} \approx 1600 \times \bar{n}_{\text{c}}^{3D}/2$, where $\bar{n}_{\text{c}}^{3D} \approx 16$ is the average number of connections per particle in 3D).

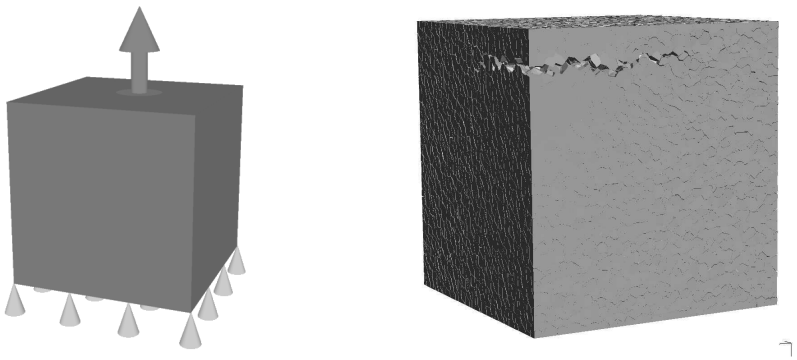

Figure 5. *Geometry and loading for the uniaxial tension test (PB), and initiation of the macrocrack*

Table 1. *Stiffness matrices features*

	number of DOF	number of nonzero elements	fill ratio
Problem 0	12 288	253 675	0.1680%
Problem A	240 000	5 010 095	0.008698%
Problem B	384 000	36 662 401	0.02486%

3. Solution to a sequence of linear systems : recycling strategies

Solving a problem with a discrete model consists then in solving a sequence of n_{rupt} systems $\mathbf{K}^k\mathbf{u}^k = \mathbf{f}^k$, $k \in \{1, ...n_{\text{rupt}}\}$ with small changes in the left hand side operator (here the stiffness matrix). The reduction of the total CPU time for the solution to such a sequence of large sparse symmetric positive definite linear systems is then a crucial point.

We explore in this paper the possibility offer by two different approaches based on the use of a krylov iterative solver (a preconditioned conjugate gradient) associated to a reuse (or a recycle) of Krylov subspaces : the generalized iterative reuse of Krylov subspaces algorithm (GIRKS) (Rey, 1996; Risler *et al.*, 2000) and the selective reuse of Krylov subspaces (SRKS) (Rey *et al.*, 1998; Gosselet *et al.*, 2002). They are based on the reuse of numerical information coming from the iterative solution to previous linear systems. They have been introduced and evaluated in association with domain decomposition strategies. Such an association induced two specificity : first a significant reduction of the size of linear problems (by condensation at the interface between subdomains) and second the used of a full re-orthogonalization procedure. In

the following, the three algorithms are introduced. Their adaptivity to the solution to a sequence of large linear systems are discussed and evaluated.

3.1. *The reference iterative solver without recycling strategy*

The preconditioned conjugate gradient is first described in Algorithm 1. For both genericity and simplicity reasons, a standard Incomplete Cholesky Preconditioner (ICP) is chosen. ICP allows to control easily the ratio performance/computing time by changing its threshold ξ. It is a quite standard choice for the iterative solution to large sparse symmetric positive definite linear systems (Saad, 2000).

Algorithm 1 PCG

$\mathbf{u}_0^k$ arbitrary
$\mathbf{g}_0^k = \mathbf{K}^k\mathbf{u}_0^k - \mathbf{f}^k$
$\mathbf{z}_0^k = \mathbf{M}^{-1}\mathbf{g}_0^k$
$i \Leftarrow 0$
repeat
  $\boldsymbol{\omega}_i^k = \mathbf{z}_i^k + \gamma_{i-1}^k \boldsymbol{\omega}_{i-1}^k$ $\qquad \gamma_{i-1}^k = -\frac{(\mathbf{z}_i^k, \mathbf{K}\boldsymbol{\omega}_{i-1}^k)}{(\mathbf{K}\boldsymbol{\omega}_{i-1}^k, \boldsymbol{\omega}_{i-1}^k)}$
  $\mathbf{u}_{i+1}^k = \mathbf{u}_i^k + \alpha_i^k \boldsymbol{\omega}_i^k$
  $\mathbf{g}_{i+1}^k = \mathbf{g}_i^k + \alpha_i^k \mathbf{K}^k \boldsymbol{\omega}_i^k$ $\qquad \alpha_i^k = -\frac{(\mathbf{g}_i^k, \mathbf{z}_i^k)}{(\mathbf{K}\boldsymbol{\omega}_i^k, \boldsymbol{\omega}_i^k)}$
  $\mathbf{z}_{i+1}^k = \mathbf{M}^{-1}\mathbf{g}_{i+1}^k$
  $i \Leftarrow i + 1$
until convergence $|\mathbf{g}_{i+1}^k| < 10^{-8}|\mathbf{g}_0^k|$

The efficiency of PCG algorithm is mainly based on the choice of an efficient preconditioner. Table 2 reports the performance of the ICP for different ξ for the problem P0. A sequence of 212 resolutions has to be performed to obtain the failure of the sample. Although the number of iterations reduces dramatically as ξ decreases, the CPU time to compute the preconditioner becomes too important with respect to total CPU time to solve the entire problem. This effect can be seen on the total CPU time to solve P0 (Table 2), and on Figure 6 where the difference of number of iterations between $\xi = 10^{-8}$ and $\xi = 10^{-9}$ (visible on the bottom graph for the first solved system) vanishes as the number of solved systems increases. The optimum is obtained for a preconditioner with a number of nonzero elements close to the number of nonzero elements of the stiffness matrix. This result is satisfactory considering that it is not reasonable to store in memory a preconditioner much larger than the initial stiffness matrix, especially for large-size problems. Next, the threshold $\xi = 10^{-8}$ will be chosen.

At last, another well-known point that may affect the efficiency of PCG is related to the numerical satisfaction of the $\boldsymbol{K}$-orthogonality of the search direction $\boldsymbol{\omega}_i^k$ with respect to previous one ($(\boldsymbol{K}\omega_i^k, \boldsymbol{\omega}_j^k) = 0, \forall j = 1, \ldots i-1$). More precisely, to reduce

Table 2. *Performance of Incomplete Cholesky preconditioner for P0*

ICP threshold ξ	1.e-7	5.e-8	2.5e-8	1.e-8	5.e-9	1.e-9
Nonzero elements	101 582	132 715	167 980	228 813	291 409	523 943
Ratio ICP size/$\mathbf{K}$ size	0.400	0.523	0.662	0.902	1.15	2.07
ICP computation CPU time (s)	8.0	9.7	11.5	14.2	17.4	29.7
Nb iterations for 1st system	138	94	67	45	36	20
Total CPU time (s)	156	146	140	131	146	165

numerical error on the $\boldsymbol{K}$-orthogonality condition, a full-reorthogonalization procedure can be used for the computation of vectors $\boldsymbol{\omega}_i^k$. Among various Gram-Schmid's procedures, one simply consists in changing Line 6 into $\boldsymbol{\omega}_i^k = \mathbf{z}_i^k + \sum_{j=0}^{i-1} \gamma_j^k \boldsymbol{\omega}_j^k$ with $\gamma_j^k = -\frac{(\mathbf{z}_i^k, \mathbf{K}\boldsymbol{\omega}_j^k)}{(\mathbf{K}\boldsymbol{\omega}_j^k, \boldsymbol{\omega}_j^k)}$. One then obtains a more robust algorithm. The drawback is obviously an extra-cost of the algorithm.

Such procedure is typically required in the context of domain decomposition method. In our case, numerical error propagation where never observed. It is an important difference with the case we deal with. Hence, such re-orthogonalization procedure is not required in our case and basic algorithm of PCG is always preferred in the following.

3.2. *First recycling strategy : GIRKS algorithm*

This strategy is based on the computation of correction terms for the preconditioned residual $\mathbf{z}_{i+1}^k$ (Algorithm 1, line 9) by reusing the r_1 vectors $\left\{\boldsymbol{\omega}_i^0\right\}_{i=1,r_1}$ generated during the iterative solution to the first problem. This correction can be seen as a non symmetric preconditioning, leading to a potentially non convergent algorithm. Hence a full re-orthogonalization of vectors $\boldsymbol{\omega}_i^k$ is required (see (Risler *et al.*, 2000) for the full algorithm). One of the advantage of the algorithm is that the resolution is still based on matrix-vector products and scalar products : several routines (BLAS, LAPACK...) can be used for optimized computations, and parallelisation of these products are easily envisageable.

For problem P0 with IC(ξ=1.e-8) preconditioner, performance in terms of iteration number of this algorithm is compared to the iteration number with PCG in Figure 7. As expected, the number of iterations per system decreases. The effect is mainly visible in

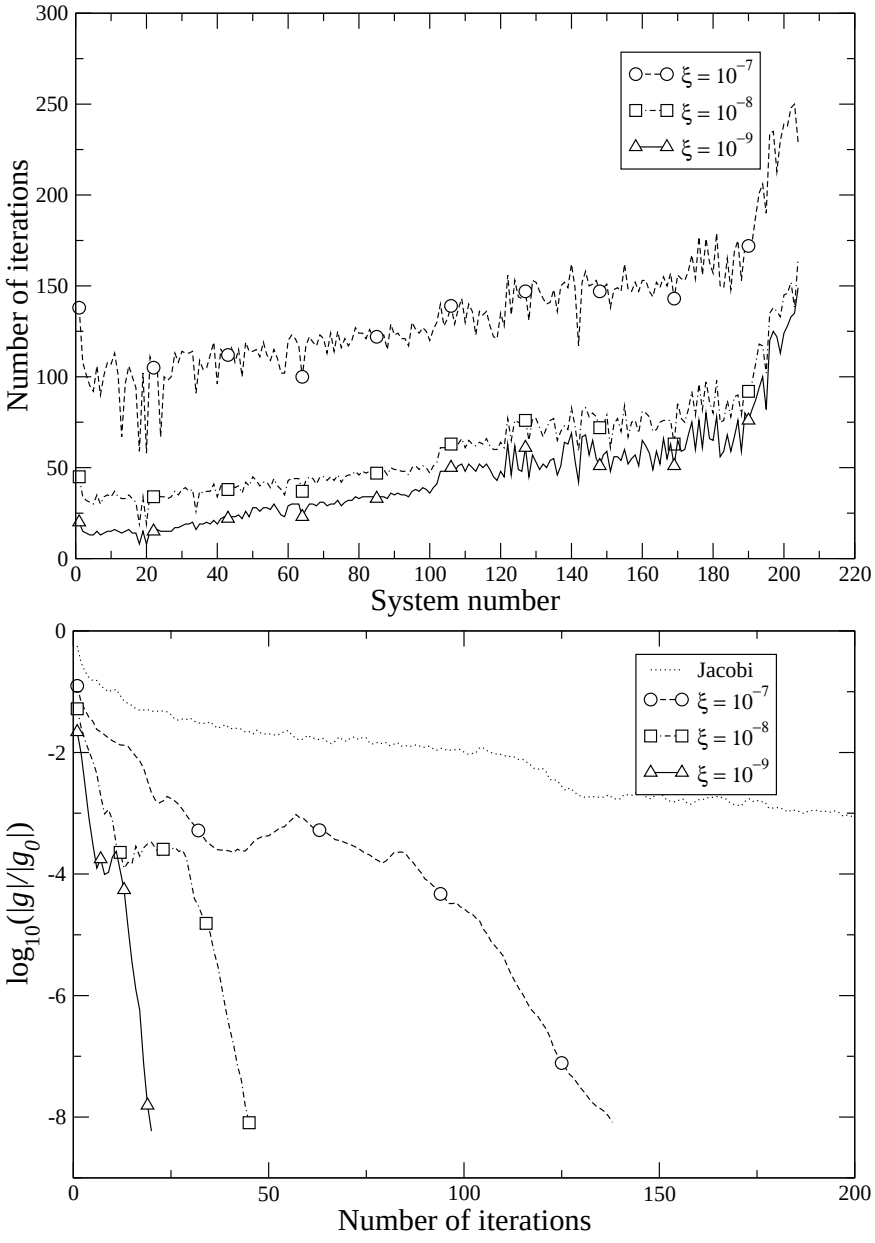

Figure 6. *P0 - Number of iterations for each system (top) and residual convergence for the first solved system (bottom) with Incomplete Choleski Preconditioner*

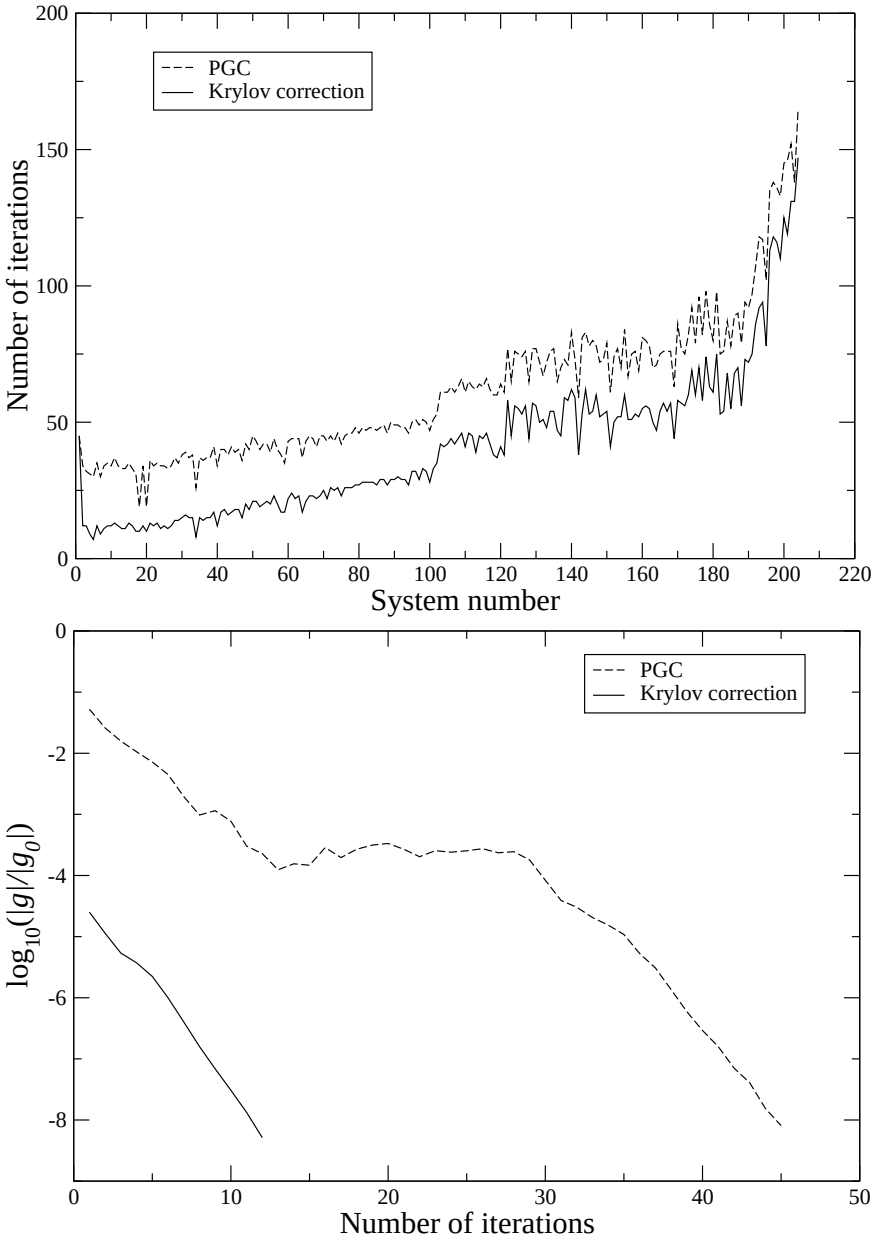

Figure 7. *Effect of Krylov correction in terms of number of iterations per system (top) and convergence rate for the second system (bottom)*

the beginning of the loading : for example, the second system needs only 12 iterations to be solved with Krylov algorithm versus 34 iterations for the PCG algorithm (the total number of iterations to solve P0 is here 8139 versus 12367 for PCG algorithm). Furthermore, the evolution of residual (Figure 7, left curve) becomes monotonic and ensures a better convergence. As the total number of solved systems increases, the gain becomes less significant : changes in stiffness matrix transform more and more the system to solve and Krylov correction becomes less relevant.

Note that improvement can be achieved by actualizing Krylov subspace during the sequence of linear systems. However we will not try this actualization. Indeed, in the context where a Gram-Schmid procedure is not required for the reference solver, such recycling strategy loses its interest if we consider the CPU time : iteration number decreases, but each iteration is more costly because of (i) the computation of the correction terms (ii) the required full re-orthogonalization procedure. The reduction in CPU time is effective (although limited) during the first solved systems, but decreases and becomes negative for the last solved systems. Such problems were not observed in the context of domain decomposition method for mainly two reasons : first it operates on a reduced-size problem and second a re-orthogonalization procedure is required on the reference iterative solver.

Anyhow, at this point, the fact is that we just have considered that stiffness matrix has limited changes between two successive resolutions, but we do not take into account that theses changes are explicitly known.

3.3. *Second recycling strategy : SRKS approach*

The selective reuse of Krylov subspaces (Rey *et al.*, 1998; Gosselet *et al.*, 2002) is the second recycling strategy we propose to evaluate. It is based on an Augmented Conjugate Gradient algorithm. The key point here is that the use of a Gram-Schmid procedure is not required and one can expect a reduction of CPU time. Furthermore with the ACG algorithm, the projector is only used once per iteration.

3.3.1. *Augmented Conjugate Gradient with SRKS strategy*

The Augmented conjugate gradient is first described in the Algorithm 2 (see (Saad *et al.*, 2000) among other).

The key point then relates on the choice of the $n_{\text{dof}}^{\text{tot}} \times r_1$ matrix $\mathbf{C}$. A natural choice is to consider r_1-Krylov space (span$\left\{\boldsymbol{\omega}_i^0,\ i = 1, \dots r_1\right\}$) generated during the solution to the first system.

In order to reduce the cost of the algorithm, one can limit the size of the matrix $\mathbf{C}$ by reducing the number of kept vectors. Then, a natural idea (Rey *et al.*, 1998) is to consider a subspace of the r_1-Krylov space. Different studies have shown that convergence of PCG algorithm is related to convergence of the Ritz vectors (der Sluis

Algorithm 2 Augmented CG

Compute $\mathbf{P} = \left[I - \mathbf{C}\left(\mathbf{C}^T\mathbf{K}^k\mathbf{C}\right)^{-1}\mathbf{C}^T\mathbf{K}^k\right]$
$\mathbf{u}_0^k = \mathbf{C}\left(\mathbf{C}^T\mathbf{K}^k\mathbf{C}\right)^{-1}\mathbf{C}^T\mathbf{f}^k$
$\mathbf{g}_0^k = \mathbf{f}^k - \mathbf{K}^k\mathbf{u}_0^k$
$\mathbf{z}_0^k = \mathbf{P}\mathbf{M}^{-1}\mathbf{g}_0^k$
$\omega_0 = z_0$
$i \Leftarrow 1$
repeat
  $\mathbf{u}_i^k = \mathbf{u}_{i-1}^k + \alpha_{i-1}^k\boldsymbol{\omega}_{i-1}^k$ $\alpha_{i-1}^k = \frac{(\mathbf{g}_{i-1}^k, \mathbf{z}_{i-1}^k)}{(\mathbf{K}^k\boldsymbol{\omega}_{i-1}^k, \boldsymbol{\omega}_{i-1}^k)}$
  $\mathbf{g}_i^k = \mathbf{g}_{i-1}^k - \alpha_{i-1}^k\mathbf{K}^k\boldsymbol{\omega}_{i-1}^k$
  $\mathbf{z}_i^k = \mathbf{P}\mathbf{M}^{-1}\mathbf{g}_i^k$
  $\boldsymbol{\omega}_i^k = \mathbf{z}_i^k + \beta_i^k\boldsymbol{\omega}_{i-1}^k$ $\beta_i^k = -\frac{(\mathbf{z}_i^k, \mathbf{K}^k\boldsymbol{\omega}_{i-1}^k)}{(\mathbf{K}^k\boldsymbol{\omega}_{i-1}^k, \boldsymbol{\omega}_{i-1}^k)}$
  $i \Leftarrow i + 1$
until convergence $|\mathbf{g}_{i+1}^k| < 10^{-8}|\mathbf{g}_0^k|$

et al., 1986; Paige *et al.*, 1995). A judicious choice for composing the matrix $\mathbf{C}$ is to keep Ritz vectors only if their associated Ritz values have a good convergence (Gosselet *et al.*, 2002). This choice is realized through the Algorithm 3 (note that superscript k has been omitted in this algorithm for clarity).

Algorithm 3 Selective Reuse of Krylov Subspaces

Define $\mathbf{V}_p = \left[\ldots, (-1)^j \frac{\mathbf{z}_j}{(\mathbf{r}_j, \mathbf{z}_j)^{1/2}}, \ldots\right]_{0 \leqslant j < p}$
Define $\mathbf{H}_p = \mathrm{tridiag}(\eta_{j-1}, \zeta_j, \eta_j)_{0 \leqslant j < p}$
  with $\zeta_0 = \frac{1}{\alpha_0}$, $\zeta_j = \frac{1}{\alpha_j} + \frac{\beta_{j-1}}{\alpha_{j-1}}$, $\eta_j = \frac{\sqrt{\beta_j}}{\alpha_j}$ (η_0 is not used)
  and $\alpha_j = \frac{(\mathbf{g}_j, \mathbf{z}_j)}{(\mathbf{K}\boldsymbol{\omega}_j, \boldsymbol{\omega}_j)}$ $\beta_j = -\frac{(\mathbf{z}_j, \mathbf{K}\boldsymbol{\omega}_{j-1})}{(\mathbf{K}\boldsymbol{\omega}_{j-1}, \boldsymbol{\omega}_{j-1})}$
Compute eigenvectors $\mathbf{q}_p$ and eigenvalues θ_p of $\mathbf{H}_p$ ($\theta_p^1 \geq \ldots \geq \theta_p^p$)
Compute $\mathbf{Y}_p = \mathbf{V}_p\mathbf{q}_p = \left[y_p^1, \ldots, y_p^p\right]$
Extract $\mathbf{H}_{p-1} = \mathrm{tridiag}(\eta_{j-1}, \zeta_j, \eta_j)_{0 \leqslant j < p-1}$ from $\mathbf{H}_p$
Compute eigenvalues $(\theta_{p-1}^1 \geq \ldots \geq \theta_{p-1}^{p-1})$ of $\mathbf{H}_{p-1}$
Keep Ritz vector $\mathbf{q}_j$ in Krylov subspace if associated eigenvalue verifies :

$$\frac{|\theta_p^j - \theta_{p-1}^{j-1}|}{\theta_p^j} \leq \delta$$

In this algorithm, δ is a constant parameter that controls the number of vectors we will kept. r_1^δ, the number of kept Ritz vectors, is equal to 0 for $\delta = 0$ or equals to r_1 if $\delta \to \infty$. With this selection, we reduce dimension of matrix $\mathbf{C}$ without redu-

cing iterative solver performance (just Ritz vectors with low convergence factor are removed).

However, as for the GIRKS algorithm, the total CPU time for this algorithm is not reduced compared to the reference PCG one. The reason is that if the full re-orthogonalization is no more needed with this algorithm, costly computation of matrix-matrix product is still performed. This kind of operations are usually avoided in iterative algorithm. Note that in the context of domain decomposition method such inefficiency in terms of CPU time were not observed mainly because of the reduced size of involved vectors (related to the interface between sub-domain).
In the next section, we introduce an evolution of this algorithm, based on the known changes of the stiffness matrix.

3.3.2. *SRKS approach for explicit changes in the left hand side operator*

In the case where the changes in the left hand side operator $\mathbf{K}^k$ are explicit, the cost of the matrix-matrix product can be reduced.

Let us consider that the bond ij breaks for step k. By using elastic prediction algorithm, we can write :

$$\mathbf{C}^T\mathbf{K}^k\mathbf{C} = \mathbf{C}^T(\mathbf{K}^{k-1} - \mathbf{L_{ij}}^T\mathbf{K_{ij}}\mathbf{L_{ij}})\mathbf{C} = \mathbf{C}^T\mathbf{K}^{k-1}\mathbf{C} - \mathbf{C}^T\mathbf{L_{ij}}^T\mathbf{K_{ij}}\mathbf{L_{ij}}\mathbf{C}$$

In the last equation, $\mathbf{C}^T\mathbf{K}^{k-1}\mathbf{C}$ is a stored $r_1 \times r_1$ matrix that has been computed during last step. $\mathbf{C}^T\mathbf{L_{ij}}^T\mathbf{K_{ij}}\mathbf{L_{ij}}\mathbf{C}$ needs only $2 \times 2n_{\text{dof}} \times r_1$ operations to be computed. This last computation and the computation of the inverse of the $r_1 \times r_1$-matrix $\mathbf{C}^T\mathbf{K}^k\mathbf{C}$ are achieved only *one time* per resolution. In the same way, the $r_1 \times n_{\text{dof}}^{\text{tot}}$-matrix $\mathbf{C}^T\mathbf{K}^k$ can be computed by using the relation :

$$\mathbf{C}^T\mathbf{K}^k = \mathbf{C}^T\mathbf{K}^{k-1} - \mathbf{C}^T\mathbf{L_{ij}}^T\mathbf{K_{ij}}\mathbf{L_{ij}}$$

By these considerations, the augmented conjugate gradient algorithm with SRKS strategy recovers strong interest : iteration number decreases, and correction computation is performed in a short time. A final remark concerns the storage of matrices. It is not reasonable to store in the computer memory the $n_{\text{dof}}^{\text{tot}} \times n_{\text{dof}}^{\text{tot}}$-matrix $\left[I - \mathbf{C}\left(\mathbf{C}^T\mathbf{K}^k\mathbf{C}\right)^{-1}\mathbf{C}^T\mathbf{K}^k\right]$. The optimized storage, in terms on number of operations, is to store $n_{\text{dof}}^{\text{tot}} \times r_1$-matrix $\mathbf{C}\left(\mathbf{C}^T\mathbf{K}^k\mathbf{C}\right)^{-1}$ and the $r_1 \times n_{\text{dof}}^{\text{tot}}$-matrix $\mathbf{C}^T\mathbf{K}^k$. We propose in the next part to evaluate the performance of this algorithm on PA and PB, both problems with a relevant number of degrees of freedom.

4. Performance of the algorithm

4.1. *Problem A*

For this problem, the failure of the beam is obtained after $n_{\text{rupt}} = 250$ rupture of bonds, so one has to solve 250 linear systems of $240\,000$ dof before failure. The threshold of the ICP is fixed to $\xi = 10^{-8}$. For the reference PCG solver, its corresponds

to 258 iterations to solve the first system, and a total of 69077 iterations for the 250 resolutions.

We present results obtained with the SRKS algorithm for different values of δ. Figure 8 shows the number of kept Krylov vectors and the total number of iterations $n_{\text{it}}^{\text{tot}}$ versus δ, and the total CPU time of the PCG algorithm compared to the SRKS algorithm.

We first observe a significant reduction of the total number of iterations, but more interesting is that the efficiency in terms of CPU time is quite constant (a reduction factor of 2.5) for a large range of $\delta \in [10^{-6}, 10^{-10}]$.

Figure 9 presents the evolution of number of iterations per system for both algorithms with $\delta = 10^{-10}$. For this problem, the reduction of the total number of iterations is really significant and is kept quasi-constant for all systems, that ensures a significant reduction of the total CPU time.

Table 3 gives the best results between the two algorithms (with $\delta = 10^{-10}$). The total number of iterations is divided by a factor 3.25 and the total CPU time is divided by a factor 2.61. The lower factor for the total time is due to the extra-cost of each iteration for the SRKS algorithm.

Table 3. *Best results for problem PA.*

		time (mn)	$n_{\text{it}}^{\text{tot}}$
PCG	258 iterations (first system)	133	69077
SRKS	14 Ritz vectors	51	21253
Ratio	18.4	2.61	3.25

4.2. *Problem B*

The failure of the cube is obtained after $n_{\text{rupt}} \approx 14000$ ruptures of bonds, so one has to solve around 14000 linear systems of $384\,000$ dof before failure. The threshold of the ICP is fixed to $\xi = 10^{-8}$. For the reference PCG solver, its corresponds to 253 iterations to solve the first system.

The analysis of the Ritz vectors threshold is just performed on the 100 first systems (100 breaking bonds). For PCG algorithm, it corresponds to 22963 iterations for these 100 systems. Like for problem A, we present results obtained with our algorithm for different values of δ. The total number of iterations $n_{\text{it}}^{\text{tot}}$ is again lower than the 22963 iterations for the PCG algorithm, and the reduction of CPU time also increases. Figure 10 shows the number of kept Krylov vectors and the total number of iterations $n_{\text{it}}^{\text{tot}}$ versus δ, and the total CPU time of the PCG algorithm compared to the SRKS algorithm. Here again, a large range of value for $\delta \in [10^{-4}, 10^{-7}]$ insures a stable factor of the reduction (greater than 2) the total CPU time. Note that the optimized

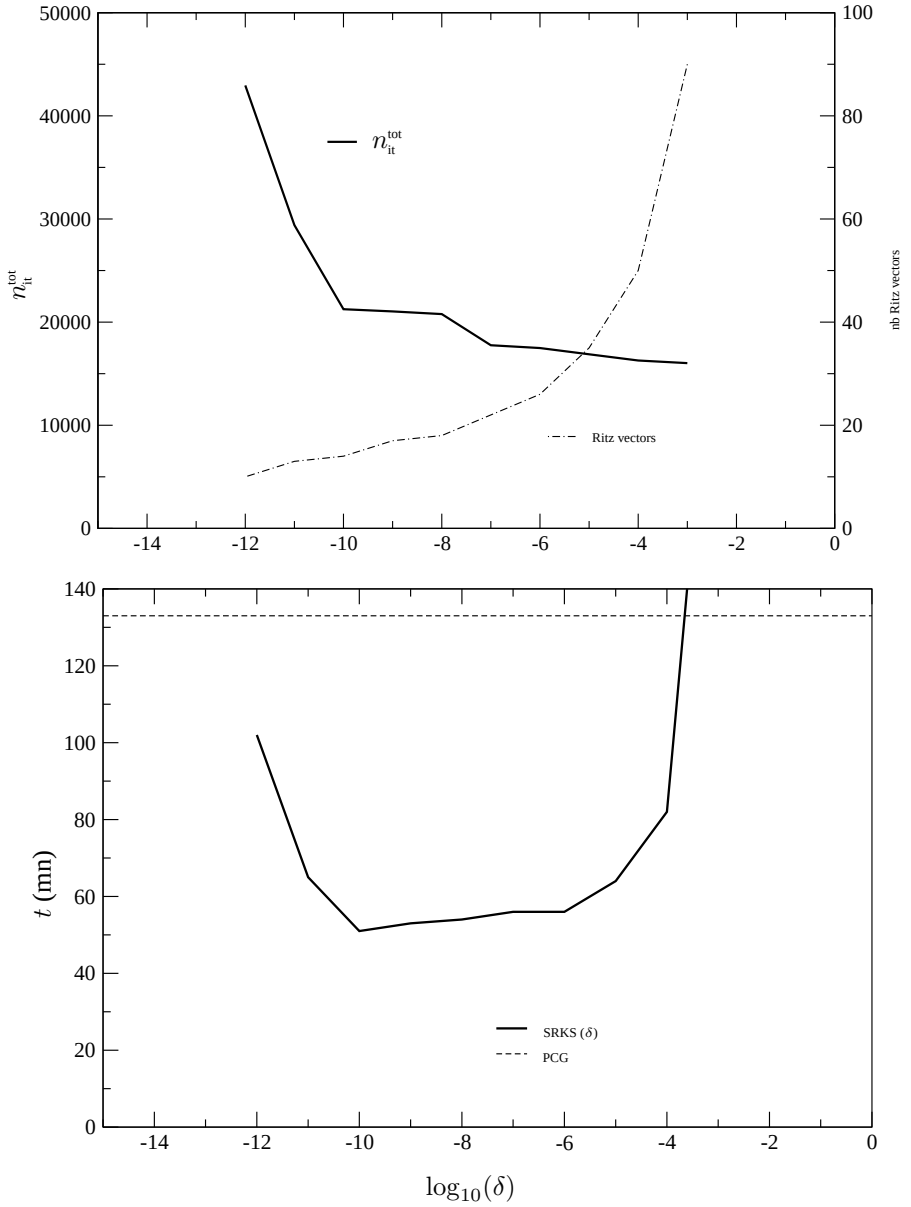

Figure 8. *PA - Number of Ritz vectors and total number of iterations versus the Ritz threshold (top) and comparison of the CPU time between SRKS and PCG algorithms (botom)*

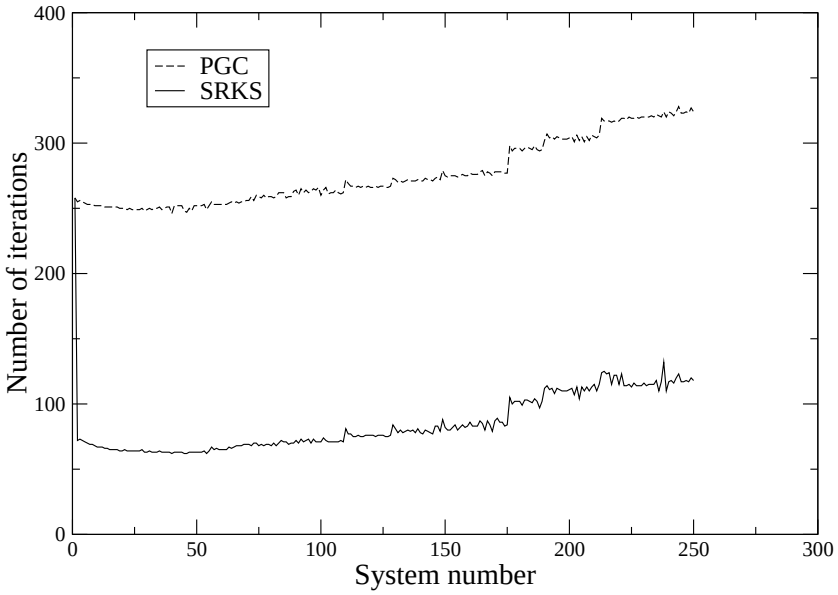

Figure 9. *PA - Evolution of number of iterations for PCG et SRKS algorithms*

range of value for δ depends on the considered numerical problem. From our experience, a value of $\delta = 10^{-7}$ is always in the range of pertinent value and can be chosen as a generic value if no specific analysis is performed.

Table 4. *Best results for problem PB*

		time (mn)	$n_{\text{it}}^{\text{tot}}$
PCG	253 iterations (first system)	383	22963
SRKS	38 Ritz vectors	174	8014
Ratio	14.1	2.20	2.87

Table 4 gives the bets results between the two algorithms for $\delta = 10^{-5}$. The total number of iterations is divided by a factor 2.87 and the total CPU time is divided by a factor 2.20.

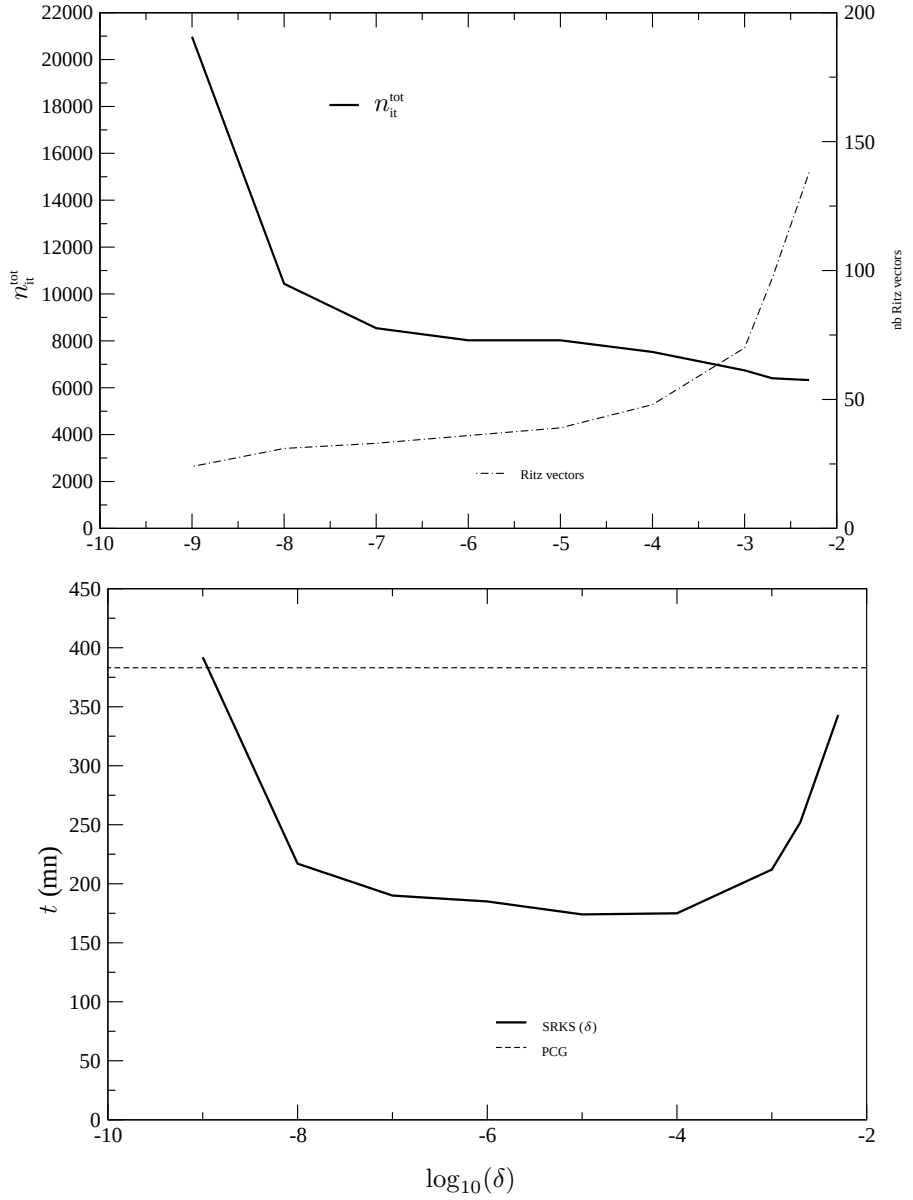

Figure 10. *PB - Number of Ritz vectors and total number of iterations versus the Ritz threshold (top) and comparison of the CPU time between SRKS and PCG algorithms (botom)*

5. Conclusion

The discrete models, based on a representation of the material with a particle assembly, are relevant to represent cracks in material. However, the large number of particles that should be considered reduces their interest in terms of computation time, especially for quasi-static problems. Then, one needs an efficient solver for an optimized resolution. In this paper, we used iterative solvers based on the recycling of Krylov subspace. For the large size systems considered in this study, we obtain as expected a significant reduction of the iteration number. On the other hand, such strategies are not efficient in term of CPU time, due to the use of a standard preconditioner (IC preconditioner in this study) instead of the optimal preconditioner coming from domain decomposition method. Anyhow, in the specific context of the discrete element approach, the explicit knowledge of the changes in the operator offers a low-cost construction of the $\mathbf{C}^T\mathbf{K}^k\mathbf{C}$ matrix, leading to a significant reduction of the CPU time. The main features of the algorithm are the following ones :

– The algorithm is not intrusive as it just requires the modification of the PCG subroutine. The modifications are based on classical matrix/vector operations.

– The over cost of the algorithm compared to a classical PCG algorithm is the storage of a $n_{\text{dof}}^{\text{tot}} \times r_1^\delta$-matrix (Krylov subspace), a $r_1^\delta \times n_{\text{dof}}^{\text{tot}}$-matrix ($\mathbf{C}^T\mathbf{K}^k$) and a small $r_1^\delta \times r_1^\delta$-marix ($\mathbf{C}^T\mathbf{K}^k\mathbf{C}$).

– The total number of iterations is expected to be divided by a factor greater than 2.5, as the CPU time is expected to be divided by a factor greater than 2.

6. References

Cundall P. A., Strack O. D. L., « A discrete numerical model for granular assemblies », *Géotechnique*, vol. 29, p. 47-65, 1979.

D'Addetta G. A., Discrete models for cohesive frictional materials, PhD thesis, Stuttgart University, 2004.

Delaplace A., « Tensile damage response from discrete element virtual testing », *Geomechanics and Geoengineering*, vol. 4, n° 1, p. 79-89, 2009.

Delaplace A., Desmorat R., « Discrete 3D model as complimentary numerical testing for anisotropic damage », *International Journal of Fracture*, vol. 148, p. 115-128, 2007.

der Sluis A. V., der Vorst H. A. V., « The rate of convergence of conjugate gradients », *Numer. Math.*, vol. 48, p. 543-560, 1986.

Farhat C., Roux F., « Implicit parallel processing in structural mechanics », *Computational Mechanics Advances*, vol. 2, p. 1-124, 1994.

Gosselet P., Rey C., « On a selective reuse of Krylov subspaces in Newton-Krylov approaches for nonlinear elasticity », *14th International Conference on Domain Decomposition Methods*, 2002.

Gosselet P., Rey C., « Non-overlapping domain decomposition methods in structural mechanics », *Arch. Comput. Meth. Engng.*, vol. 13, p. 515-572, 2006.

Herrmann H. J., Roux S., *Statistical models for the fracture of disordered media*, Elsevier Science Publishers, Amsterdam, 1990.

Moukarzel C., Herrmann H. J., « A vectorizable random lattice », *J. Stat. Phys.*, vol. 68, p. 911-923, 1992.

Nukala P., Simunovic S., « An efficient block-circulant preconditioner for simulating fracture using large fuse networks », *J. Phys. A : Math. Gen.*, vol. 37, p. 2093-2103, 2004.

Nukala P., Simunovic S., « Large scale simulation of fracture network », *ICF 11 - 11th International Conference on Fracture*, 2005.

Paige C., Parlett B. N., der Vorst H. V., « Approximate solutions and eigenvalue bounds from Krylov subspaces », *Num. Lin. Alg. Appl.*, vol. 2, p. 115-133, 1995.

Potyondy D. O., Cundall P. A., « A bonded-particle model for rock », *International Journal of Rock Mechanics and Mining Sciences*, vol. 41, n° 8, p. 1329-1364, 2004.

Rey C., « An acceleration technique for the solution of non-linear elasticity problems by domain decomposition », *C. R. Acad. Sci.*, vol. 322, p. 601-606, 1996.

Rey C., Risler C., « A Rayleigh-Ritz preconditioner for the iterative solution to large scale nonlinear problem », *Numerical Algorithms*, vol. 17, p. 279-311, 1998.

Risler C., Rey C., « Iterative accelerating algorithms with Krylov subspaces for the solution to large-scale nonlinear problems », *Numerical Algorithms*, vol. 23, p. 1-30, 2000.

Saad Y., *Iterative methods for sparse linear systems*, PWS Publishing Company, 3rd edition, 2000.

Saad Y., Yeung M., Erhel J., Guyomarc'h F., « A deflated version of the conjugate gradient algorithm », *SIAM J. Sci. Comput.*, vol. 21, p. 1909-1926, 2000.

Van Mier J. G. M., Van Vliet M. R. A., Wang T. K., « Fracture mechanisms in particle composites : statistical aspects in lattice type analysis », *Mech. Mater*, vol. 34, p. 705-724, 2002.

Yip M., Li Z., Liao B.-S., Bolander J., « Irregular lattice models of fracture of multiphase particulate materials », *International Journal of Fracture*, vol. 140, p. 113-124, 2006.

Received : 2 October 2009
Accepted : 17 November 2009