

---

# Solveurs parallèles pour la simulation de systèmes multicontacts

**Mathieu Renouf — Pierre Alart**

*LMGC, UMR 5508, Université Montpellier 2- CNRS cc048,  
place Eugène (Bataillon), F-34095 Montpellier cedex 5  
renouf@lmgc.univ-montp2.fr*

---

*RÉSUMÉ. Dans le cadre des systèmes multicontacts, et en particulier des milieux granulaires, la méthode de dynamique des contacts (Non Smooth Contact Dynamics) intègre un solveur de type Gauss Seidel non linéaire à chaque pas de temps. Nous proposons ici deux implémentations de cet algorithme aux comportements différents dans leur version parallèle. Un nouvel algorithme de type Gradient Conjugué, intrinsèquement parallèle, est également présenté.*

*ABSTRACT. In the context of multicontact systems, especially in the case of granular media, the Non Smooth Contact Dynamics method uses a Non Linear Gauss-Seidel algorithm at each time step to solve the contact problem. We present two kinds of implementation and their influence on multithreading. A new Conjugate Gradient type algorithm is introduced.*

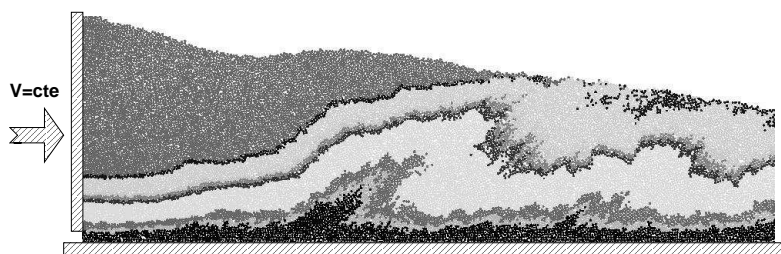
*MOTS-CLÉS : contact, NSCD, parallèle, gradient conjugué.*

*KEYWORDS: contact, NSCD, multithreading, conjugate gradient.*

---

## 1. Motivations

L'utilisation du calcul parallèle pour la simulation de systèmes multicontacts tels que les milieux granulaires s'avère un moyen efficace pour garder des temps de simulation raisonnables devant des modélisations numériques de plus en plus importantes en nombre d'éléments et de plus en plus réalistes. Pour donner un ordre d'idée, la reproduction bidimensionnelle d'une simulation analogique de boîte à sable (utilisée par les géophysiciens [KUK 02] pour reproduire et comprendre l'évolution des plis de la croûte terrestre) de dimensions  $50 \times 3$  centimètres (largeur  $\times$  hauteur) nécessite plus de 60 000 particules.



**Figure 1.** Modélisation numérique réduite d'une boîte à sable : échantillon de 40 000 particules

Des techniques de calcul parallèle dédiées aux éléments discrets ont déjà été expérimentées, toutes basées sur la décomposition de domaine et la distribution des tâches [BRE 99, OWE 00]. Nous n'exploiterons pas cette approche ici, excluant toute sous-structuration géométrique difficile à mettre en œuvre tout au long d'un processus. Les résultats présentés sont issus de simulations réalisées avec la plate forme logicielle *LMGC90* ("Logiciel de Mécanique Gérant le Contact", écrit en Fortran90), code dédié à la simulation de problèmes de contacts, développé par M. Jean et F. Dubois [DUB 03]. Une analyse de la consommation du temps *CPU*, pour différentes géométries et dynamiques, permet de distinguer les parties du code consommatrices en temps *CPU*. Il s'est avéré que pour les systèmes "denses" (compactage et tambour), où la notion de système multicontact trouve pleinement sa signification, le solveur est le plus grand consommateur de temps de calcul (*cf.* tableau 1). C'est la raison pour laquelle notre principal effort s'est d'abord porté sur le solveur.

Après avoir effectué un rappel sur la méthode et l'algorithme utilisés dans la section 2, nous présentons et comparons les deux types d'implémentation mis en place (section 3) et leur comportement face à une exécution parallèle (section 4). Toujours en vue d'une optimisation du temps de résolution, un nouvel algorithme intrinsèquement parallèle basé sur une approche de type Gradient Conjugué est introduit dans la section 5 ; les premiers tests comparatifs sont présentés.

problème	solveur	convergence	détection
brassage ( $e = 1$ )	18,6	2,9	47,4
compactage	84,7	2,4	5,8
tambour tournant	85,7	2,5	1,9

**Tableau 1.** Répartition du temps écoulé dans différentes parties du code ( % )(e : coefficient de restitution)

## 2. Dynamique non régulière

La démarche utilisée par M. Jean et J.-J Moreau, et appelée *Non Smooth Contact Dynamics* [JEA 99, MOR 93], préserve le caractère non régulier lié aux actions intergranulaires régies par des lois dont les graphes possèdent des pentes “raides” [CAM 01] (choc, condition unilatérale, frottement de Coulomb). Les équations de la dynamique linéarisées écrites sur l’intervalle  $]t^i, t^{i+1}]$  pour la configuration  $\mathbf{q}$  du système, privilégient la variable vitesse  $\dot{\mathbf{q}}$  comme inconnue cinématique,

$$\mathbf{M}^i(\dot{\mathbf{q}}^{i+1} - \dot{\mathbf{q}}^i) = h\mathbf{R}_{free}^i + \mathbf{R}^{i+1}, \quad [1]$$

où  $i$  désigne l’indice du pas de temps,  $\mathbf{M}^i$  représente la matrice de masse du système (composantes de masse et d’inertie),  $\mathbf{R}_{free}^i$  les efforts extérieurs sans prise en compte des efforts de contact,  $\mathbf{R}^{i+1}$  l’impulsion de contact moyenne, inconnue dans un schéma implicite, et  $h$  le pas de temps. La matrice de masse étant aisément inversible (matrice diagonale), nous pouvons réécrire l’équation (1) de la façon suivante,

$$\dot{\mathbf{q}}^{i+1} = \dot{\mathbf{q}}_{free}^i + (\mathbf{M}^{-1})^i \mathbf{R}^{i+1}, \text{ où } \dot{\mathbf{q}}_{free}^i = \dot{\mathbf{q}}^i + h(\mathbf{M}^{-1})^i \mathbf{R}_{free}^i \quad [2]$$

où  $\dot{\mathbf{q}}_{free}^i$  désigne la “vitesse libre” *i.e.* la vitesse calculée avec les seuls efforts extérieurs. Le problème d’interaction est résolu au niveau local (contacts) en termes de variables locales :  $\mathbf{v}_\alpha$ , vitesse relative au contact entre deux particules, et  $\mathbf{r}_\alpha$  impulsion au contact ( $\alpha$  désigne le numéro du contact). Ce niveau est préféré au niveau global, celui des corps. Dans la configuration locale notre système devient

$$\mathbf{v}_\alpha^{i+1} = \mathbf{v}_{\alpha,free}^i + \sum_{\beta=1}^{n_c} \mathbb{W}_{\alpha\beta} \mathbf{r}_\beta^{i+1}, \quad [3]$$

où  $\mathbb{W}_{\alpha\beta} = \mathbb{H}_\alpha^t(\mathbf{q}^i)(\mathbf{M}^{-1})^i \mathbb{H}_\beta(\mathbf{q}^i)$  ( $2 \times 2$  en 2D et  $3 \times 3$  en 3D) et  $\mathbf{v}_{\alpha,free}^i = \mathbb{H}_\alpha^t(\mathbf{q}^i)\dot{\mathbf{q}}_{free}^i$ ,  $\mathbb{H}^\alpha$  étant une application linéaire passant du repère local au repère global. Nous sommes alors amené à résoudre contact par contact le système suivant (4)

conformément à une stratégie de type Gauss Seidel non linéaire par blocs (Non Linear Gauss Seidel - NLGS),

$$\begin{cases} \mathbf{v}_\alpha^{k+1} - \mathbb{W}_{\alpha\alpha}\mathbf{r}_\alpha^{k+1} = \mathbf{v}_{\alpha,free} + \sum_{\beta<\alpha} \mathbb{W}_{\alpha\beta}\mathbf{r}_\beta^{k+1} + \sum_{\beta>\alpha} \mathbb{W}_{\alpha\beta}\mathbf{r}_\beta^k \\ LoiContact_\alpha[\mathbf{v}_\alpha^{k+1}, \mathbf{r}_\alpha^{k+1}] = true \end{cases} . [4]$$

Pour alléger les notations, l'indice temporel  $i$  est omis ; l'indice  $k$  est relié aux itérations. Le terme  $LoiContact[.,.]$  décrit les lois de contact frottant intervenant localement (Loi de Signorini et Coulomb). Pour des problèmes bidimensionnels, il peut être résolu par intersection de graphe affine. Dans le cas de problèmes tridimensionnels, nous utilisons une méthode de Newton généralisée [ALA 97].

### 3. Implémentation

L'implémentation de l'algorithme NLGS au sein du code peut se faire de deux façons différentes. La première méthode consiste à ne construire que les matrices blocs  $\mathbb{W}_{\alpha\alpha}$  sur la diagonale. Le calcul du second membre dans la première équation de (4), noté  $\mathbf{b}_\alpha$  par la suite, ne s'effectue pas à l'aide des matrices  $\mathbb{W}_{\alpha\beta}$  mais par des échanges entre les niveaux local (contacts) et global (grains) *via* les opérateurs  $\mathbb{H}_\alpha$ ,  $\mathbb{H}_\alpha^t$  et les torseurs des efforts appliqués sur chaque solide (noté  $\mathbf{R}_j$  pour le solide  $j$ ), et ce avant et après la résolution du système local (4). Cette méthode sera appelée ELG (*Echange Local Global*). La seconde méthode consiste à calculer au préalable toutes les matrices non nulles  $\mathbb{W}_{\alpha\beta}$ . L'information pour calculer le terme  $\mathbf{b}_\alpha$  est alors stockée au niveau local, et aucun échange entre les deux niveaux n'est nécessaire. Nous appellerons cette méthode SDL (*Stockage Données Locales*). La méthode ELG minimise la charge mémoire mais augmente le nombre d'opérations à effectuer. Par contre la phase de préparation de la méthode SDL est plus coûteuse que celle de la méthode ELG. Le tableau 2 représente les différentes étapes de l'algorithme de la méthode ELG. C'est lors des phases (i) et (iv) du tableau 2 que les passages local-global sont effectués. Lors de la phase (i), pour le contact  $\alpha$  nous descendons chercher des informations au niveau des torseurs des deux particules en contact pour pouvoir exprimer le second membre  $\mathbf{b}_\alpha$ . La phase (iv) consiste à actualiser les torseurs des particules précédentes à partir de la nouvelle impulsion locale calculée. Le tableau 3 représente l'algorithme de la méthode SDL. Les étapes (i) et (iv) du tableau 2 sont supprimées. Elles sont prises en compte dans l'étape de préparation lors du calcul des matrices  $\mathbb{W}_{\alpha\beta}$ . L'étape (i) est alors remplacée par une somme de produits matrice vecteur. Ce sont par ailleurs les seules différences entre les deux algorithmes.

Une étude comparative à été faite entre les deux méthodes. Nous avons réalisé différentes simulations faisant varier le nombre et la taille des grains pour représenter un processus identique (dépôt sous gravité). Le nombre de contacts est de 1 070, 4 100, 8 200, 17 100, 33 500, 70 400 et 137 040. Les deux techniques d'implémentation sont testés sur chaque échantillon. La figure 2 représente l'évolution du rapport de diffé-

<p>nouveau pas de temps</p> <p>Evaluation de <math>\dot{\mathbf{q}}_{free}</math> ainsi que <math>\mathbf{v}_{\alpha,free}</math> (<math>\alpha = 1, nc</math>)</p> <p>Calcul des matrices blocs diagonales <math>\mathbb{W}_{\alpha\alpha}</math></p> <p><math>k = k + 1</math> (iteration NLGS)</p> <p><math>\alpha = \alpha + 1</math> (indice de contact)</p> <p>(i) Calcul de <math>\mathbf{v}_{\alpha,aux}</math> en identifiant les deux corps en contact :  <math>\alpha = jl</math> et <math>\mathbf{v}_{\alpha,aux} = \mathbb{H}_{\alpha}^t [(\mathbf{M}_j)^{-1} \mathbf{R}_j^k - (\mathbf{M}_l)^{-1} \mathbf{R}_l^k]</math></p> <p>(ii) Evaluation de <math>\mathbf{b}_{\alpha}</math> (membre de droite)</p> <p><math>\mathbf{b}_{\alpha} = \mathbf{v}_{\alpha,free} + \mathbf{v}_{\alpha,aux} - \mathbb{W}_{\alpha\alpha} \mathbf{r}_{\alpha}^k</math></p> <p>(iii) Résolution du problème local avec (4),  les inconnues étant <math>(\mathbf{v}_{\alpha}^{k+1}, \mathbf{r}_{\alpha}^{k+1})</math>.</p> <p>(iv) Actualisation de <math>\mathbf{R}_j</math> et de <math>\mathbf{R}_l</math></p> $\begin{bmatrix} \mathbf{R}_j \\ \mathbf{R}_l \end{bmatrix}^{k+1} = \begin{bmatrix} \mathbf{R}_j \\ \mathbf{R}_l \end{bmatrix}^k + \mathbb{H}_{\alpha} (\mathbf{r}_{\alpha}^{k+1} - \mathbf{r}_{\alpha}^k)$ <p>Test de convergence pour <math>k = 0 \dots k_{max}</math></p> <p>Evaluation de <math>\dot{\mathbf{q}}^{i+1}</math> grace à (2).</p>
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

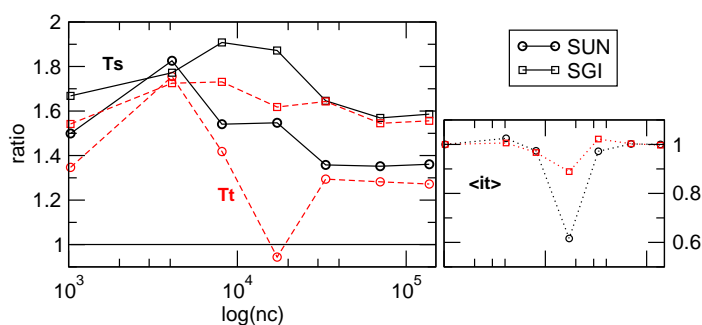
Tableau 2. Algorithme de la méthode ELG

<p>nouveau pas de temps</p> <p>Evaluation de <math>\dot{\mathbf{q}}_{free}</math> ainsi que <math>\mathbf{v}_{\alpha,free}</math> (<math>\alpha = 1, nc</math>)</p> <p>Calcul des matrices blocs <math>\mathbb{W}_{\alpha\beta}</math></p> <p><math>k = k + 1</math> (iteration NLGS)</p> <p><math>\alpha = \alpha + 1</math> (indice de contact)</p> <p>(i) Evaluation de <math>\mathbf{b}_{\alpha}</math> (membre de droite)</p> <p><math>\mathbf{b}_{\alpha} = -\mathbf{v}_{\alpha,free} - \sum_{\beta &lt; \alpha} \mathbb{W}_{\alpha\beta} \mathbf{r}_{\alpha}^{k+1} - \sum_{\beta &gt; \alpha} \mathbb{W}_{\alpha\beta} \mathbf{r}_{\alpha}^k</math></p> <p>(ii) Résolution du problème local avec (4),  les inconnues étant <math>(\mathbf{v}_{\alpha}^{k+1}, \mathbf{r}_{\alpha}^{k+1})</math>.</p> <p>Test de convergence pour <math>k = 0 \dots k_{max}</math></p> <p>Evaluation de <math>\dot{\mathbf{q}}^{i+1}</math> grace à (2).</p>
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Tableau 3. Algorithme de la méthode SDL

rentes grandeurs en fonction du  $\log_{10}$  du nombre de contact  $n_c$  :  $T_s$  correspond au temps d'une itération ELG sur le temps d'une itération SDL,  $T_t$  correspond au temps total de la méthode ELG sur celui de la méthode SDL et  $\langle it \rangle$  le nombre d'itérations ELG moyen sur le nombre d'itérations SDL moyen (encadré droit de la figure 2). Les simulations ont été réalisés sur deux types de calculateurs (*SUN* et *SGI*) afin d'évaluer l'influence de l'architecture machine.

Pour les deux types d'architecture, la méthode SDL reste plus performante que la méthode ELG sur l'ensemble des simulations ( $T_s$  et  $T_t$  restent presque toujours supérieurs à 1). Toutefois on peut noter une différence de valeur pour  $T_s$  suivant que l'on soit sous *SUN* ou *SGI*, les valeurs obtenues sous *SGI* sont supérieures à celles obtenues sous *SUN*. Pour les simulations effectuées sous *SUN*, pour de petits échantillons ( $n_c < 4000$ ), la méthode SDL apparaît comme étant environ 1,5 fois plus rapide que la méthode ELG. Ce rapport diminue pour des échantillons plus conséquents ( $n_c > 100\,000$ ) pour atteindre une valeur de 1,3. Sous *SGI*, le rapport  $T_s$  varie moins et reste proche de 1,6 avec un pic proche de 2 pour des valeurs de  $n_c$  voisines de 10 000. Si l'on observe le rapport du nombre d'itérations moyen des méthodes ELG et SDL, on s'aperçoit qu'il est relativement constant : il peut subir toutefois des crises (cas de l'échantillon à 17 100 contacts sous *SUN* où le solveur a du mal à converger), ce qui se répercute ensuite sur le temps total de la simulation (graphe  $ratio(\log(nc))$ ). Les différences entre les performances obtenues sous *SUN* et *SGI* sont liées à l'espace mémoire disponible ;  $T_s$  étant défini comme un rapport de temps de calcul, la vitesse des processeurs n'intervient pas. L'espace utilisé étant plus important sous *SGI*, la méthode SDL est privilégiée (il n'en est pas ainsi sous *SUN*) justifiant l'écart entre les courbes de la figure 2.



**Figure 2.** Rapport du temps des simulations ELG et SDL en fonction du  $\log_{10}$  du nombre de contacts :  $T_s$  est le ratio du temps relatif passé dans le solveur,  $T_t$  est le ratio du temps total. En insert, rapport du nombre moyen d'itérations obtenu avec les deux techniques d'implémentation

Sur la gamme d'étude, la méthode SDL apparaît donc comme étant la plus efficace. Toutefois l'évolution du ratio nous laisse supposer que pour des échantillons encore plus conséquents ( $10^5$ ,  $10^6$  particules, ...) les deux méthodes pourraient devenir équivalentes, voire la méthode ELG pourrait devenir plus performante que la méthode SDL. Notons que ce constat est bien entendu tributaire de la machine utilisée puisque le facteur pénalisant pour la méthode SDL est la place mémoire. Il faut aussi noter que l'étape de préparation SDL est plus coûteuse que celle ELG. Ainsi à temps équivalent dans la partie solveur, il devient plus intéressant d'utiliser l'implémentation ELG.

#### 4. Multithreading

Le solveur Gauss Seidel non linéaire est par nature séquentiel (résolution d'un contact après l'autre), contrairement à un solveur de type Jacobi par exemple (résolution simultanée de tous les contacts). Cet aspect a été abordé dans [REN 04b] indépendamment du choix de la programmation, ELG ou SDL ; on a montré que d'un point de vue algorithmique le nombre d'itérations nécessaire à la convergence n'était que faiblement affecté par la parallélisation. Plus précisément, sur la base d'une programmation ELG, un découpage "simple" de la boucle des contacts, *via* des directives OpenMP utilisables par des ordinateurs multi-processeurs à mémoire partagée, entraîne une légère perturbation du nombre d'itérations mais également fournit des solutions différentes à chaque lancement du code, du fait de la renumérotation engendrée par le découpage en *threads* et à la multiplicité des solutions du problème mécanique. Il engendre aussi des accès simultanés à la mémoire machine par différents processeurs (*Race Condition*) ; l'une des valeurs n'est alors pas prise en compte. Tout ceci ne nuit toutefois pas à la qualité macroscopique de la solution, et permet de retrouver les mêmes grandeurs physiques (compacité, anisotropie) [TRO 02, RAD 98]. L'application du *multithreading* à la version SDL diffère en ce point de la méthode précédente : le phénomène de *Race Condition* est écarté puisque les échanges avec le niveau global sont supprimés lors du processus itératif. Pour apprécier l'efficacité des codes parallèles, nous avons évalué le *speed-up* relatif défini par la formule suivante :

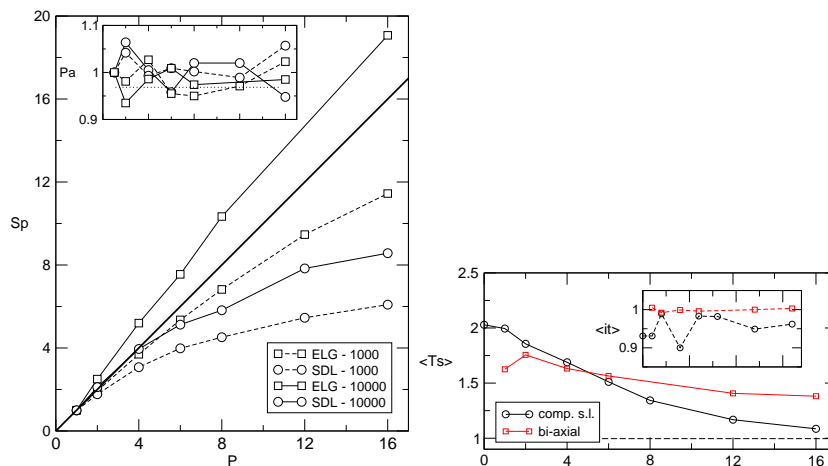
$$S_P = p_a^P \frac{T_{seq}}{T_P} \quad \text{où} \quad p_a^P = \frac{\langle It \rangle_{seq}}{\langle It \rangle_P}, \quad [5]$$

où  $T_P$ , respectivement  $\langle It \rangle_P$ , représente le temps de calcul, respectivement le nombre d'itérations moyen,  $P$  désignant le nombre de processeurs utilisés (l'indice *seq* désignant un calcul séquentiel) ;  $p_a^P$  représente la performance algorithmique. En d'autres termes,  $S_P$  représente le rapport des temps de calcul d'une itération séquentielle sur une itération parallèle.

REMARQUE.— Les grandeurs indicées par  $P = 1$  et *seq* sont différentes. En effet, il s'avère que l'utilisation d'un code compilé avec des directives OpenMP utilisé en monoprocesseur prend plus de temps qu'un code compilé sans directives OpenMP ; le rapport des temps de calcul pouvant varier, suivant l'architecture et la taille de l'échantillon, de 1,05 à 1,2.

Les différents graphes de la figure 3 fournissent l'évolution de ces grandeurs suivant le nombre de processeurs utilisés. Les tests ont été réalisés sur un calculateur *SGI Origin 3 800* à processeurs *R14 000/500Mhz*. Les courbes de la partie gauche de la figure 3 illustrent l'évolution de  $S_p$  pour les deux méthodes. L'implémentation ELG possède un meilleur comportement en parallèle que la méthode SDL, la courbe se rapprochant de la droite de comportement optimal (droite de pente 1) pour ELG et s'en écartant pour SDL. L'augmentation de la taille des échantillons contribue à l'amélioration du *speed-up* relatif comme montré dans [REN 04b] observé ici pour les deux méthodes. La partie droite de la figure 3, vient en compléter la figure 2, en

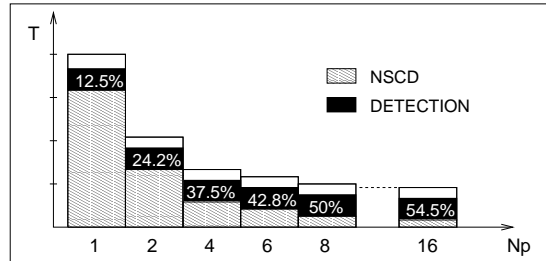
comparant les deux techniques d'implémentation lors d'une exécution parallèle sur des échantillons de 10 000 particules soumis à un compactage à surface libre et à un essai bi-axial. On retrouve ainsi le comportement observé sur la figure 2 pour une exécution séquentielle, à savoir un temps plus raisonnable avec l'implémentation SDL observé quel que soit le nombre de processeurs. Ce gain en temps voisin de 2 pour une exécution séquentielle chute avec l'augmentation du nombre de processeurs pour atteindre un rapport proche de 1, 1 avec 16 processeurs. On peut observer également grâce à l'insert gauche de la figure 3, que la performance algorithmique reste proche de 1, n'engendrant ainsi pas de grandes différences entre les deux techniques d'implémentation au niveau du nombre d'itérations moyen (insert droit de la figure 3). L'allure des courbes laisse supposer que pour une utilisation parallèle massive ( $P > 100$ ) l'implémentation ELG sera préférable à l'implémentation SDL.



**Figure 3.** A gauche : Speed-Up relatif en fonction du nombre de processeurs pour les deux techniques d'implémentation. A droite : rapport des temps d'une itération ELG et SDL en fonction du nombre de processeurs. En inserts : évolution de la performance algorithmique (haut-gauche) et rapport du nombre d'itérations moyen ELG/SDL (bas-droite)

La figure 4 met en évidence l'évolution du coût du solveur dans le coût global en fonction du nombre de processeurs. En particulier la part de la détection des contacts – elle aussi parallélisable – devient prépondérante. Ainsi pour une simulation sur 16 processeurs, la partie occupée par le solveur est minimale, et dans ce cadre-là, passer à 32 processeurs serait sans intérêt. La parallélisation de la détection est donc primordiale pour espérer avoir un code parallèle performant. On pourra bénéficier à ce titre de méthodes de détection de contacts efficaces pré existantes et déjà parallélisées [CHR 03].





**Figure 4.** Comparaison du temps passé dans la détection des contacts et dans le solveur lors d'une utilisation parallèle

## 5. Gradient projeté conjugué

### 5.1. Formulation

Pour les problèmes linéaires, l'algorithme de type Gradient Conjugué est plus rapide que Gauss-Seidel pour peu qu'il soit préconditionné, et est de plus intrinsèquement parallèle (produits matrice-vecteur) mais son comportement sur des problèmes non linéaires utilisant des projections reste à déterminer.

Deux difficultés sont à surmonter. La première concerne les propriétés "imparfaites" de la matrice  $\mathbb{W}$ , qui tout en étant symétrique n'est généralement que semi-définie positive, singularité traduisant l'indétermination de systèmes de collections denses de rigides. Mais si cette propriété ne facilite pas l'analyse mathématique de la convergence des méthodes itératives [CAO 02], elle n'est cependant pas un obstacle irrémédiable à la convergence pratique des méthodes itératives ; en témoigne le bon comportement de *NSCD* [MOR 99]. La deuxième difficulté provient du couplage du contact unilatéral et du frottement de Coulomb qui ne permet plus de formuler un problème variationnel classique. Nous optons cependant pour écrire un problème de quasi-optimisation [ALA 91] contraint où le convexe  $\mathcal{C}$  des contraintes dépend de la solution elle-même  $\mathcal{C} = \mathcal{C}(\mathbf{r})$  (on parle aussi d'inéquation quasi variationnelle)

$$\mathbf{r} = \underset{\mathbf{s} \in \mathcal{C}(\mathbf{r})}{\operatorname{argmin}} \frac{1}{2} \mathbf{s} \cdot \mathbb{W} \mathbf{s} - \mathbf{b} \cdot \mathbf{s}. \quad [6]$$

Pour des systèmes bidimensionnels, cadre dans lequel nous resterons,  $\mathcal{C}$  reste un polyèdre convexe. On peut classifier les méthodes de gradients sur des problèmes d'optimisation sous contraintes en introduisant successivement des améliorations : projection de l'itéré, projection du gradient (Rosen), conjugaison [REN 04a]. En définitive, l'al-

gorithme du gradient projeté conjugué (Conjugate Projected Gradient - CPG) s'écrit alors

$$\mathbf{r}^{k+1} = \text{proj}_{\mathcal{C}} \{ \mathbf{r}^k + \alpha^k \{ \text{proj}(\mathbf{u}^k; T_{\mathcal{C}}(\mathbf{r}^k)) + \beta^k \text{proj}(\mathbf{p}^{k-1}; T_{\mathcal{C}}(\mathbf{r}^k)) \} \} \quad [7]$$

L'adaptation au gradient conjugué s'effectue en projetant sur le cône simultanément le résidu  $\mathbf{u}^k$  et le gradient précédent  $\mathbf{p}^{k-1}$  avant de procéder à la "conjugaison" qui s'effectue alors dans le cône tangent en  $\mathbf{r}^k$  à  $\mathcal{C}$  par le calcul de  $\beta^k$  puis  $\mathbf{p}^k$  (cf. tableau 4).

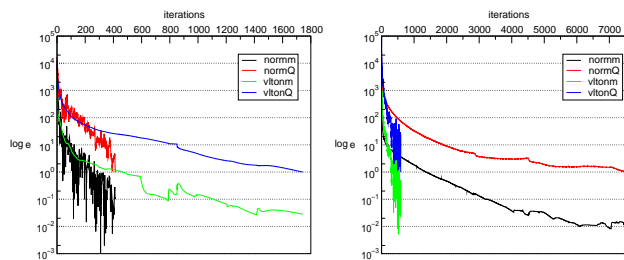
Calcul de $\mathbf{u}^0 = \mathbf{b} - \mathbb{W}\mathbf{r}^0$ Initialisation de $\mathbf{w}^0 = \mathbf{u}^0$ , $\mathbf{p}^0 = \mathbf{u}^0$ $k = k + 1$ (iteration GPC)
$\alpha^{k+1} = \frac{\mathbf{u}^k \cdot \mathbf{p}^k}{\mathbf{p}^k \cdot \mathbb{W}\mathbf{p}^k}$
$\mathbf{r}^{k+\frac{1}{2}} = \mathbf{r}^k + \alpha^{k+1} \mathbf{p}^k$
$\mathbf{r}^{k+1} = \text{proj}_{\mathcal{C}}(\mathbf{r}^{k+\frac{1}{2}})$
$\mathbf{u}^{k+1} = \mathbf{b} - \mathbb{W}\mathbf{r}^{k+1}$
$\mathbf{w}^{k+1} = \text{proj}(\mathbf{u}^{k+1}; T_{\mathcal{C}}(\mathbf{r}^{k+1}))$
$\mathbf{z}^{k+1} = \text{proj}(\mathbf{p}^k; T_{\mathcal{C}}(\mathbf{r}^{k+1}))$
$\beta^{k+1} = -\frac{\mathbf{w}^{k+1} \cdot \mathbb{W}\mathbf{p}^k}{\mathbf{p}^k \cdot \mathbb{W}\mathbf{p}^k}$
$\mathbf{p}^{k+1} = \mathbf{w}^{k+1} + \beta^{k+1} \mathbf{z}^{k+1}$
Test de convergence jusqu'à $k_{max}$ fixé

**Tableau 4.** Algorithme du gradient projeté conjugué (CPG)

L'algorithme proposé diffère donc du gradient conjugué linéaire par les seules projections, de l'itéré sur le convexe, des gradients sur le cône. Ces projections sont locales (*i.e.* par contact) et indépendantes, donc parallélisables, tout comme les produits scalaires et matrice-vecteur. La conjugaison des gradients joue pleinement son rôle dans l'efficacité de la méthode tant que le cône tangent (ensemble des contraintes actives) ne change pas, en particulier quand il coïncide avec celui de la solution. Or nous avons remarqué que l'algorithme NLGS peine à converger y compris quand le statut solution (contraintes actives) a été trouvé. Sur les itérations précédant ce statut, on ne peut présumer de l'efficacité de notre méthode, la conjugaison n'opérant pas. On trouvera dans [MAY 86, DIL 88] des approches similaires pour le seul contact unilatéral. Une extension consistante au contact frottant nécessite des développements non détaillés dans cet article et que l'on peut trouver dans [REN 04a].

## 5.2. Tests numériques

Nous présentons ici les premiers résultats numériques réalisés avec la méthode CPG. Nous avons déposé sous pesanteur des disques dans une boîte, et une fois l'échantillon au repos, procédé à une rotation de  $30^\circ$  instantanée ; le coefficient de frottement grain-grain et grain-paroi est respectivement de 0,3 et 0,5.



**Figure 5.** Comparaison des convergences des deux méthodes sur des échantillons de 32 disques (gauche) et 750 disques (droite) : Les courbes de convergences du NLGS sont lisses mais très lentes à atteindre le critère contrairement aux courbes du CPG qui converge rapidement mais de façon plus erratique

Sur ces premiers tests, la convergence du CPG, quoique chahutée, est beaucoup plus rapide que NLGS. Ces premiers résultats encourageants nous ont poussé à poursuivre plus loin les simulations. On trouvera dans [REN 04a] une étude comparative plus détaillée.

## 6. Conclusion

La première version parallèle de la méthode *NSCD* donne des résultats intéressants, notamment avec la méthode *ELG*. La programmation *SDL* fournit un code présentant un moins bon potentiel en exploitation parallèle. Le stockage de  $\mathbb{W}$  doit être amélioré afin d'y remédier car cette approche présente l'avantage d'encapsuler le solveur et donc de le rendre indépendant du contexte d'utilisation, ici milieux granulaires. Les premiers résultats obtenus avec la méthode CPG en séquentiel sont très positifs et le Gradient Projeté Conjugué constitue donc une alternative intrinsèquement parallèle à la méthode de Gauss Seidel non linéaire.

## 7. Bibliographie

- [ALA 91] ALART P., CURNIER A., « A mixed formulation for frictionnal contact problems prone to Newton like solution methods », *Comp. Meth. Appl. Mech. Engrg.*, vol. 92, 1991, p. 353-375.

- [ALA 97] ALART P., « Méthode de Newton Généralisée en Mécanique du contact », *J. Math. Pures Appl.*, vol. 76, 1997, p. 83-108.
- [BRE 99] BREITKOPF P., JEAN M., « Modélisation parallèle des matériaux granulaires », *Actes du quatrième colloque national en calcul des structures - Giens*, vol. 1, CSMA-AFM, 1999, p. 387-392.
- [CAM 01] CAMBOU B., JEAN M., *Micromécanique des matériaux granulaires*, Hermès Science, 2001.
- [CAO 02] CAO Z.-H., « On the convergence of iterative methods for solving singular linear systems. », *J. Comput. Appl. Math.*, vol. 145, 2002, p. 1-9.
- [CHR 03] CHRISOCHOIDES N., NAVE D., « Parallel Delaunay mesh generation kernel », *Int. J. Numer. Meth. Engng*, vol. 58, 2003, p. 161-176.
- [DIL 88] DILINTAS G., LAURENT-GENGOUX P., TRYSTRAM D., « A conjugate projected gradient method with preconditioning for unilateral contact problems. », *Comp. Struct.*, vol. 29, n° 4, 1988, p. 675-680.
- [DUB 03] DUBOIS F., JEAN M., « LMGC90 une plateforme de développement dédiée à la modélisation des problèmes d'interaction », *Actes du sixième colloque national en calcul des structures*, vol. 1, CSMA-AFM-LMS, 2003, p. 111-118.
- [JEA 99] JEAN M., « The non-smooth contact dynamics method », *Comp. Meth. Appl. Mech. Engrg*, vol. 177, 1999, p. 235-257.
- [KUK 02] KUKOWSKI N., LALLEMAND S., MALAVIEILLE J., GUTSCHER M.-A., RESTON T., « Mechanical decoupling and basal duplex formation observed in sandbox experiments with application to the Western Mediterranean Ridge accretionary complex », *Mar. Geol.*, vol. 186, 2002, p. 29-42.
- [MAY 86] MAY H.-O., « The conjugate gradient method for unilateral problems. », *Comp. Struct.*, vol. 12, n° 4, 1986, p. 595-598.
- [MOR 93] MOREAU J.-J., « New computation methods in granular dynamics. », THORNTON, Ed., *Powder & Grains*, Balkema Press, 1993, p. 227-232.
- [MOR 99] MOREAU J., « Numerical aspects of sweeping process », *Comp. Meth. Appl. Mech. Engrg*, vol. 177, 1999, p. 329-349.
- [OWE 00] OWEN D.R.J. FENG Y.T. H. K., PERIC, « Dynamic domain decomposition and load balancing in parallel simulation of finite/discrete elements », *European Congress on Computational Methods in Applied sciences and Engineering*, ECCOMAS, CD-ROM, 2000.
- [RAD 98] RADJAI F., WOLF D. E., JEAN M., MOREAU J.-J., « Bimodal Character of Stress Transmission in Granular Packings », *Phys. Rev. Lett.*, vol. 80, n° 1, 1998, p. 61-64.
- [REN 04a] RENOUF M., ALART P., « Conjugate gradient type algorithms for frictional multicontact problems : applications to granular materials », to appear in *Comp. Meth. Appl. Mech. Engrg.*, 2004.
- [REN 04b] RENOUF M., DUBOIS F., ALART P., « A parallel version of the Non Smooth Contact Dynamics Algorithm applied to the simulation of granular media », to appear in *Comput. Appl. Math.*, vol. 168, 2004, p. 375-382.
- [TRO 02] TROADEC H., RADJAI F., ROUX S., CHARMET J., « Model for granular texture with steric exclusion », *Phys. Rev. E.*, vol. 66, 2002.