A parallel version of the Non Smooth Contact Dynamics Algorithm applied to the simulation of granular media

Mathieu Renouf, Frédéric Dubois, Pierre Alart

A parallel version of the non smooth contact dynamics algorithm applied to the simulation of granular media

Mathieu Renouf*, Frédéric Dubois, Pierre Alart

Laboratoire de Mécanique et Génie Civil, Equipe Système Multi Contact, UM2, cc048 , Place Eugène Bataillon, Montpellier Cedex 05, 34095, France

Abstract

The NSCD method has shown its efficiency in the simulation of granular media. Since the number of particles and contact increases, the shape of the discrete elements becomes more complicated and the simulated problems becomes more complex, the numerical tools need to be improved in order to preserve reasonable elapsed CPU time. In this paper we present a parallelization approach of the NSCD algorithm and we investigate its influence on the numerical behaviour of the method. We illustrate the efficiency on an example made of hard disks: a free surface compaction.

Keywords: Granular material; Nonsmooth contact dynamic; Parallel; OpenMP

1. Introduction

The present work deals with the simulation of granular media which concern a wide range of practical engineering applications. One can find many examples as concrete, monuments, geomaterials (blocky rocks), powders (composites, grains, etc.), etc. All these materials are composed of particles between which local mechanical interactions define the behaviour of the medium at a macroscopic scale.

The development and the improvement of numerical methods devoted to the simulation of multi-body contact problem is of great interest and the NSCD method has shown its efficiency in this area [9,10].

* Corresponding author. Tel.: 33-467144984; fax: 33-467143923.
E-mail address: renouf@lmgc.univ-montp2.fr (M. Renouf).

This implicit time stepping method relates to a nonlinear Gauss–Seidel like algorithm, and differs from the widely used smoothed time-stepping approach [5] and from the event driven ones [6].

Because the number of particles (n_p) increases (up to 40 000) and therefore the number of contacts increases ($n_c \propto 2 * n_p$ in 2D and $n_c \propto 3 * n_p$ in 3D), or the shapes of discrete elements may be more complicated (polygons or polyhedrons), or the simulated processes more complex, the tools need to be improved in order to preserve reasonable elapsed CPU time.

An extreme example is the railway ballast fatigue simulation. To be realistic it needs up to 30 000 polyhedrons and about 1 million of loading cycles. Each loading cycle needs 1000 time steps which takes about 2 h of CPU time. Nevertheless in this work, we will present results on examples involving a moderate number of discrete elements.

In this way, parallel computation is one possibility. Various experiences have been made for the simulation of granular material, but all are based on domain decomposition methods. For example Jean et al. [3] used a static geometrical domain decomposition method (using the NSCD algorithm) or Owen et al. [8] used a topological dynamic domain decomposition method based on a smooth discrete element method approach (DEM). All these works have shown that a correct load balancing is difficult to perform.

As a matter of fact it is important to keep in mind that, in simulation of granular media, all computational efforts come from the interaction computation (contact detection and contact behaviour), which have an erratic nature even in quasistatic simulations. This is quite different from problems involving deformable bodies [1] where the computational effort comes from volumic behaviour computations which stay globally constant.

Therefore our approach will be quite different, and is encouraged by the availability of shared memory computers. It consists in parallelizing the NSCD algorithm itself, independently of any geometric or topologic information. Technically this is performed using OpenMP (http://www.openmp.org [7]) directives. It presents major advantages: its use is transparent, and its implementation allows to keep the same source code for parallel or scalar use.

2. CPU time analysis

The first part of this work consists in identifying the main CPU time consuming portions of the code (as contact detection, contact solver). But this identification strongly depends on geometry and intrinsic property of the sample, and we show a relationship between the CPU time consuming rates of the different parts of the code and the mechanical properties of the sample. The granular media can be considered as a gaz (mixing), as a liquid (avalanche, rotative drum, granular flows) or a solid (quasi-static evolution, compaction, shear test) according to the process. Three different examples have been chosen to illustrate the principal fields of application, such as, a mixing, a free surface compaction and a rotative drum. Each simulation takes into account 1000 “poly-disperse” disks, with elastic shocks for mixing and an inelastic shocks for the other simulations [4]. Fig. 1 shows the contact network in each case. This network does not exist in the mixing case, because of the permanent agitation of the material: each particle move in ballistic flight between two impacts. Nevertheless, it is more important in the two other cases which involve dense material. The percentage of elapsed time given in the Table 1 confirms this argument: the solver of the nonlinear

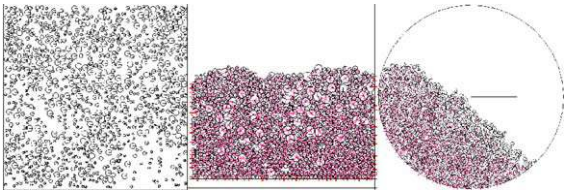

Fig. 1. Three characteristic examples (with contact network).

Table 1
Repartition of elapsed time taken by subroutine (%)

Problem	Solver (%)	Convergence (%)	Detection (%)
Mixing	18.6	2.9	47.44
Compaction	84.68	2.43	5.82
Drum	85.68	2.52	1.89

contact equations consumes the major part of the CPU time for the two last examples, although the detection of the pairs is the more expansive for the mixing case.

Consequently to this observation the programming effort is carried out on the solver itself. Therefore the sample tests considered in the following is only compaction (preferred to drum). The mixing does not fall into our priority.

3. Non smooth contact dynamics

3.1. Method description

The starting point of the method is the dynamic equations. After linearization it is written as

$$\mathbb{M}^i(\dot{\mathbf{q}}^{i+1} - \dot{\mathbf{q}}^i) = \mathbb{R}^i_{\text{free}} + h\mathbb{R}^{i+1}, \tag{1}$$

where i denotes the time step number, $\mathbb{M}^i$ is the matrix of the system (mass and inertial components), q the configuration parameter, $\mathbb{R}^i_{\text{free}}$ the residue omitting contact reactions, $h\mathbb{R}^{i+1}$ the mean contact impulsions and h the time step. Since the mass matrix is easily invertible, we can re-write Eq. (1) as

$$\dot{\mathbf{q}}^{i+1} = \dot{\mathbf{q}}^i_{\text{free}} + (\mathbb{M}^{-1})^i h\mathbb{R}^{i+1}, \quad \text{where } \dot{\mathbf{q}}^i_{\text{free}} = \dot{\mathbf{q}}^i + (\mathbb{M}^{-1})^i \mathbb{R}^i_{\text{free}}. \tag{2}$$

According to the definition of $\mathbb{R}^i_{\text{free}}$, $\dot{\mathbf{q}}^i_{\text{free}}$ notes the "free velocity".

The interaction problem is solved at the local level, and our equations need to be written in terms of local variables: $\mathbf{v}^\alpha$ the relative velocity, $\mathbf{r}^\alpha$ the contact impulse (α denotes the contact number). One obtains after

$$(\mathbf{v}^\alpha)^{i+1} = (\mathbf{v}^\alpha_{\text{free}})^i + h\sum_{\beta=1}^{nc} w^{\alpha\beta}(\mathbf{r}^\beta)^{i+1}, \tag{3}$$

where $w^{\alpha\beta} = \mathbb{H}^{\alpha *}(\mathbf{q}^i)(\mathbb{M}^{-1})^i \mathbb{H}^{\beta}(\mathbf{q}^i)$ and $(\mathbf{v}^{\alpha}_{\text{free}})^i = \mathbb{H}^{\alpha *}(\mathbf{q}^i)\dot{\mathbf{q}}^i_{\text{free}}$. The $w^{\alpha\beta}$ computation can be made with standard global condensation, or block standard global condensation, or with a numerical condensation (for rigid collection). $\mathbb{H}$ denotes a linear mapping from the local frame to the global one.

The local solution is made through a contact-by-contact like nonlinear Gauss–Seidel method. So we consider the contact α and suppose that the others are fixed: the index i of time increment is omitted. The iterative scheme is defined as follows (iteration $k+1$):

$$(\mathbf{v}^{\alpha})^{k+1} - hw^{\alpha\alpha}(\mathbf{r}^{\alpha})^{k+1} = (\mathbf{v}^{\alpha}_{\text{free}}) + h\sum_{\beta<\alpha} w^{\alpha\beta}(\mathbf{r}^{\beta})^{k+1} + h\sum_{\beta>\alpha} w^{\alpha\beta}(\mathbf{r}^{\beta})^{k},$$

$$\text{law}_{\alpha}[(\mathbf{v}^{\alpha})^{k+1}, (\mathbf{r}^{\alpha})^{k+1}] = \text{true}. \tag{4}$$

The local solution of this problem consists in finding the couple $(\mathbf{v}^{\alpha}, \mathbf{r}^{\alpha})$ satisfying Eq. (4), where $\text{law}_{\alpha}[(\mathbf{v}^{\alpha})^{k+1}, (\mathbf{r}^{\alpha})^{k+1}] = \text{true}$ expresses the frictional contact law (Signorini–Coulomb law) at the local level has to be satisfied. The right-hand side of the first equation in (4) is noted $\mathbf{b}^{\alpha}$. In a bi-dimensional description, it can be solved by looking for an affine graph intersection. In a tri-dimensional description we must use a generalized Newton method as explained by Alart and Curnier [2]. Scheme of the solver

- $i = i + 1$ (time step)
- Evaluating $\dot{\mathbf{q}}^i_{\text{free}}$ then $\mathbf{v}^{i,\alpha}_{\text{free}}$ $(\alpha = 1, nc)$
  - $k = k + 1$ (NSCD iteration)
    - $\alpha = \alpha + 1$ (contacts index)
    - Evaluating $\mathbf{b}^{\alpha}$ (right hand side)
    - Solving the local problem, unknowns $(\mathbf{v}^{\alpha}, \mathbf{r}^{\alpha})$ (via (4))
  - Convergence test
- Evaluating $\dot{\mathbf{q}}^{i+1}$ (using (2))

3.2. Multithreading procedure

Since NSCD is a nonlinear Gauss–Seidel method with a sequential structure, it is not a priori well suited for a parallel treatment. Indeed a blind parallelization of the algorithm modifies the course of the operations regarding the contact scanning order and to memory access conflicts. A preliminary study on the linear Gauss–Seidel method [11] seems to show a weak influence of the contact scanning order on the numerical behaviour of the method.

The parallelization scheme consists in splitting the contact loop between P threads which may be related to different processors (multithreading procedure). This method which leads to a contact loop renumbering, can generate a *race condition* which may be reduced with a weak bandwidth of the matrix $\mathbb{W}$ (assembled from the $w^{\alpha\beta}$ block matrix). The following numerical study consists in evaluating the influence of the multithreading on the efficiency of the NSCD method in terms of the number of iterations but also in terms of the quality of the solution.

4. Results

4.1. Sample information

In this part we will discuss the results obtained on different free surface compaction problem.

We put 1016 poly-disperse disks under gravity (for 95% average radius equal to 0.01 m and 5% equal to 0.02 m) in a box.

For all disks, the mass coefficient is 580 kg m^{-3}. After the depot, the velocity of the lateral walls is governed by the following law:

$$|v_x| = \frac{1}{60}\left(1 - \cos\left(\frac{\pi t}{30}\right)\right).$$

The process is performed using 10 000 time steps ($h = 6 \cdot 10^{-2}$ s). The compaction is performed considering three situations. The first one (DAF) uses a friction coefficient equal to 0.3 (0.5 with walls). The second (DSF) is a frictionless case. In this two cases, each walls of the box is modelized with a single rigid body. The last situation (DOSF) involves a zero friction coefficient and the walls of the box are described with a large collection of fixed disks.

4.2. Performance analysis

Time simulations are given in Table 2. Table 2 shows that the NSCD solver is slightly perturbed by the parallelization. Indeed, the number of extra iterations does not exceed 12% of the sequential iterations number. In some cases the iterations number may decrease, specially when friction occurs. When we use a collection of disks to modelize the wall of the box, the iterations number is more stable. It may be related to the smaller bandwidth of the matrix $\mathbb{W}$ in this case, although the contacts number is bigger.

To appreciate the efficiency of the parallel software, we have to evaluate, in addition to the iterations number, the computer performance related to the parallel architecture (here shared memory) and the OpenMP directives. The Code runs on a SUNFIRE 880 with six UltraSparc III (750 MHz) processors. We use a relative speed-up for P processors, S_P, as follows:

$$S_P = \frac{T_{\text{seq}}}{\text{It}_{\text{seq}}} \frac{\text{It}_P}{T_P}, \tag{5}$$

Table 2
Performance analysis for different simulations with 1,2,4 and 6 processors

Processors problem	1		2		4		6	
	TS	⟨it.⟩	TS	σ	TS	σ	TS	σ
DAF	21531	335	10535	−6	5425	−3	3874	−9
DSF	28722	382	14174	+15	7782	+48	5780	+54
DOSF	37000	441	18064	+1	9483	−3	6945	+5

TS gives the CPU time elapsed in the solver, ⟨it.⟩ represents the average iteration number of the sequential computing, whereas, σ is the gain or loss of average iteration number of the parallel computing with respect to ⟨it.⟩.

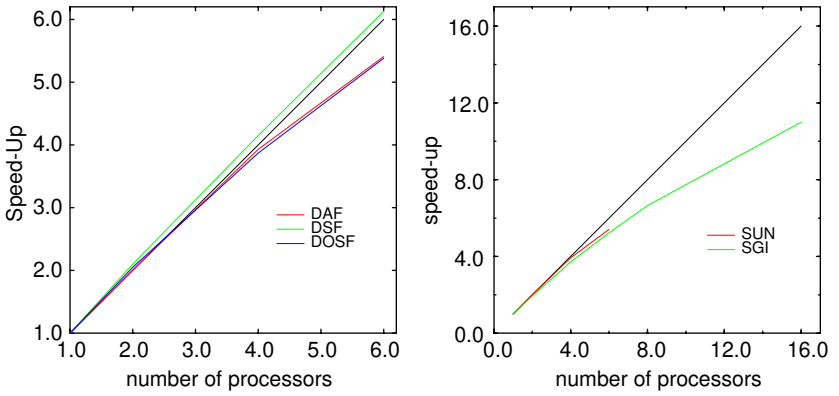

Fig. 2. Speed-up for different simulations on the same computer (left) and for a simulation on two different computers (right).

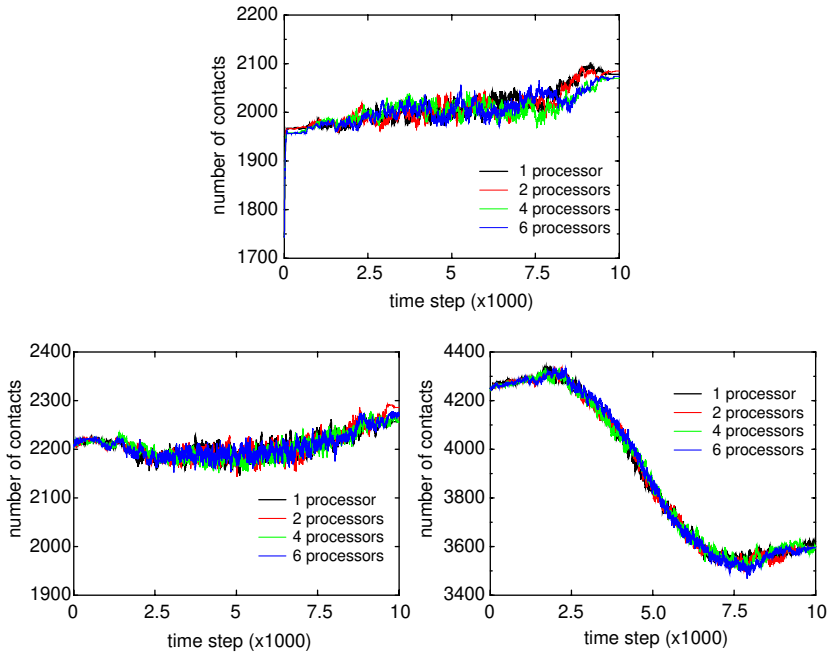

Fig. 3. Contacts number evolution for DAF (up), DSF (down-left) and DOSF (down-right).

where T_P and It_P represent, respectively, the computing time and the iterations number with P processors ($T_{\text{seq}} = T_1$). If $S_P = P$, the parallelization is efficient. Fig. 2 shows the evolution of S_P versus P for the test problems. We can see some values greater than one. This phenomenon, called *superlinear* behaviour, could be due to the activity of the computer during the computing times, to memory effect or to optimized compilation. The main result of Fig. 3 is the decrease of the performance when the number of processor increases. But this decrease is not too strong and similarly on different computer (*SUNFIRE 880* and *SGI origin 3800*—CINES France). We can then

think that the method stays still efficient with more processors for simulation with a larger number of particles.

4.3. Influence on the solution

It is interesting to see how do the solutions differ. The solutions obtained with different number of processors are not always identical in terms of the distribution of the big grains and of the local contact network. The evolution of granular medium is an erratic process with multiple possible paths; at each time step the distribution of the contact forces does not have a unique solution. The way the contact loop is performed determines one solution among several admissible ones. However, we can compare the solutions at the macroscopic level of the whole granular medium by considering the evolution of the total contacts number and the distribution of normal contact orientations (fabric tensor). Even if the contacts number in the different computations is not the same, its evolution is similar in each case. We can observe that the fluctuation of contacts number increases with the velocity (Fig. 3) that is to say in the middle of the computation where the velocity is maximal. On the other hand in a quasi-static evolution, these perturbations should be reduced.

In frictionless simulations, more stable than frictional ones, the normal contact orientations for sequential and parallel computing keep the same characteristics (cf. Figs. 4 and 5) where the directions 0°, 60° and 120° are preponderant; this texture is related to the way we prepare the sample before the process. We can note the same property with frictional simulations, but it is less pronounced (cf. Fig. 6).

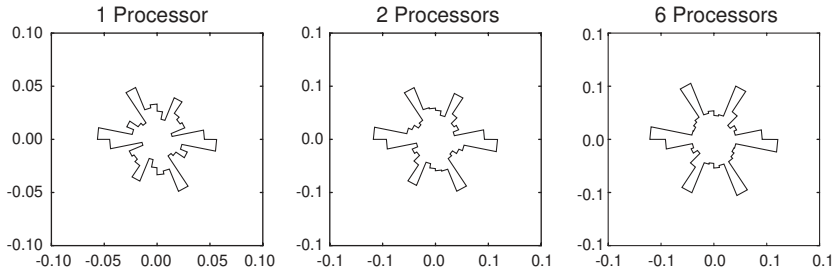

Fig. 4. Normal contact orientations for DSF simulations (1, 2 and 6 processors).

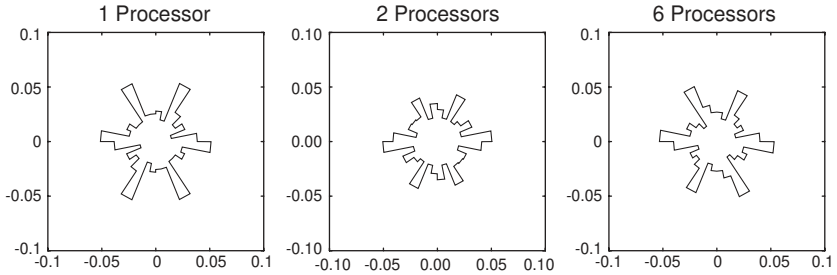

Fig. 5. Normal contact orientations for DOSF simulations (1, 2 and 6 processors).

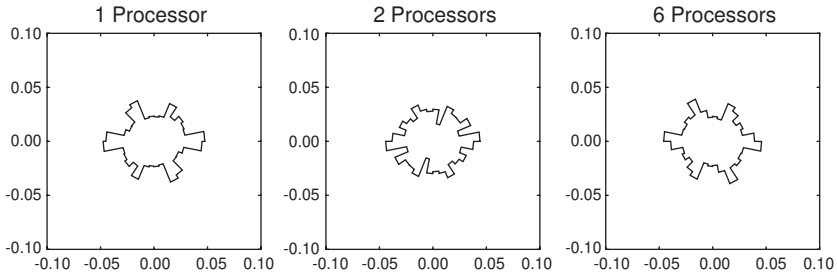

Fig. 6. Normal contact orientations for DAF simulations (1, 2 and 6 processors).

5. Conclusion

In the case of evolution of dense granular media, the parallelization method seems to give concordant macroscopic results with sequential one and gives reasonable elapsed times. In general, the principal problem is to find macroscopic comparison criterions (less difficult in quasi-static evolution than in granular flow), so we look for benchmarks in order to validate the method.

A parallel development of other portions of the code is in progress, but are intrinsically parallel as the contacts detection. Larger tests should be performed to show if the conclusions obtained from this preliminary results are influenced by the size of the samples.

We hope to use other implementation of the method and other algorithms, more efficient and less sensitive to parallelization than Gauss–Seidel method, in order to increase the size of our samples.

References

[1] P. Alart, M. Barboteu, P. Le Tallec, M. Vidrascu, An iterative schwartz method for non symmetric problems, in: N. Debit, M. Garbey, R. Hoppe, J. Périaux, D. Keyes, Y. Kuznetsov (Eds.), Thirteenth International Conference on Domain Decomposition Methods, Lyon, France; http://www.ddm.org, 2000.

[2] P. Alart, A. Curnier, A mixed formulation for frictional contact problems prone to newton like solution methods, Comput. Methods Appl. Mech. Eng. 92 (1991) 353–375.

[3] P. Breitkopf, M. Jean, Modélisation parallèle des matériaux granulaires, in: 4ème colloque national en calcul des structures, Giens, 1999.

[4] B. Cambou, M. Jean, Micromécanique des matériaux granulaires, Hermès Science, Paris, 2001.

[5] P. Cundall, A computer model for simulating progressive large scale movements of blocky rock systems, in: Proceedings of the Symposium of the International Society of Rock mechanics, Nancy, France, Vol. 1, 1971, pp. 132–150.

[6] C. Glocker, F. Pfeiffer, Multibody Dynamics with Unilateral Contacts, Wiley, New York, 1996.

[7] E. Gondet, P.-F. Lavallee, Cours OpenMP, IDRIS, Paris, 2000.

[8] K. Han, D.R.J. Owen, Y.T. Feng, D. Peric, Dynamic domain decomposition and load balancing in parallel simulation of finite/discrete elements, in: European Congress on Computational Methods in Applied Sciences and Engineering, ECCOMAS, Barcelona, Spain, 2000.

[9] M. Jean, The non-smooth contact dynamics method, Comput. Methods Appl. Mech. Eng. 177 (1999) 235–257.

[10] M. Jean, J.J. Moreau, Unilaterality and dry friction in the dynamics of rigid bodies collection, in: A. Curnier (Ed.), Contact Mechanics International Symposium, Lausanne, Switzerland, Presses Polytechniques et Universitaires Romanes, 1992, pp. 31–48.

[11] M. Renouf, Méthode de dynamique du contact et calcul parallèle, Master's Thesis, Université Montpellier II, 2001.