Methods of parallel computation applied on granular simulations

Gustavo H. B. Martins[1,⋆] and *Allbens P. F.* Atman[1,2,3,⋆⋆]

[1] *Programa de Pós-Graduação em Modelagem Matemática e Computacional - PPG-MMC, Centro Federal de Educação Tecnológica de Minas Gerais, CEFET–MG, Av. Amazonas 7675, 30510-000, Belo Horizonte, MG, Brazil.*
[2] *Physics and Mathematics Departament, Centro Federal de Educação Tecnológica de Minas Gerais, CEFET–MG, Av. Amazonas 7675, 30510-000, Belo Horizonte, MG, Brazil.*
[3] *Instituto Nacional de Ciência e Tecnologia de Sistemas Complexos (INCT-SC).*

Abstract. Every year, parallel computing has becoming cheaper and more accessible. As consequence, applications were spreading over all research areas. Granular materials is a promising area for parallel computing. To prove this statement we study the impact of parallel computing in simulations of the BNE (Brazil Nut Effect). This property is due the remarkable arising of an intruder confined to a granular media when vertically shaken against gravity. By means of DEM (Discrete Element Methods) simulations, we study the code performance testing different methods to improve clock time. A comparison between serial and parallel algorithms, using OpenMP® is also shown. The best improvement was obtained by optimizing the function that find contacts using Verlet's cells.

1 Introduction

Granular materials have high relevance in Nature as well in humans activities like in mining, in food industry, in construction technology [3–9]. Basically, granular materials are everywhere, from sand to snow, from iron ore pellets to corn grains, from dust to stones. This ubiquity justify the high importance to known their behavior in order to use and manipulate them under different situations.

Several phenomena displayed by granular materials are quite surprising, and in general is a very difficult task to predict the behavior of a substantial quantity of granular material subjected to external loading, a very common situation. Perform experiments under controlled environment is the most desired approach to acquire knowledge about these systems, but most of time only with a detailed modeling of the system is possible to verify theoretical hypothesis. Thus, it is essential to develop a reliable computational model able to predict and reproduce granular behavior under different situations, and increasing system sizes.

Given the importance and presence of granular materials in the world, simulations offer lots of advantages for testing possible characteristic states, to predict behaviors, to save money in execution of projects, and to help plan next moves in engineering for instance. But some disadvantages may be present also, as the increasing of computation time for large systems sizes. With the increasing number of agents in simulation, higher is the computation time. Most cases, this increasing relation is quadratic in time because the nature of operations.

To improve simulations, and save time, more than one machine can execute the tasks at same time. Parallelization can solve single parts of the problem separately and get them together as part of the final solution, saving total operation time. Also, many researchers are using parallel computation resources in scientific programs to simulate DEM in particles systems [1, 2]. These DEMs are commonly used to simulate different kind of systems, like dense granular materials.

Is possible to perform parallelization in clusters of pc's or multicore computers, since they have more than one CPU (Central Processing Unit) available, that exchange information between them. In the model we use, some operations can be done separately, each one in one processor. More processors used, generally result in lower total execution time from a given job. The increasing number of cores in processors nowadays has turned this possibility very accessible. Also, GPUs (Graphics Processing Units) are becoming a cheaper option to improve a lot computational power.

Given all these possibilities to improve simulations, we propose to measure different techniques to simulate a paradigmatic example of granular system, the Brazil Nut Effect (BNE), and compare the performance of the algorithms. The BNE is a typical segregation phenomena observed in granular materials, which still have several open issues to be explored by simulations. We present results of BNE with frictional and frictionless walls.

In section 2, the methodology to simulate the system is shown, with different proposals to improve the time spent

[⋆] e-mail: gmartins@adm.cefetmg.br
[⋆⋆] e-mail: atman@dppg.cefetmg.br

on serial and parallel algorithms, and some results. In section 3, the BNE results using these codes. In section 4, the main conclusions we got from our experiments.

2 Methods to save time in simulations

To simulate granular systems computationally, some models can be used [10]. One of this models considers rigid grains which are allowed to display some degree of interpenetrations, which are used in Kelvin-Voight rheology model, Cundal-Strack elastic contact in normal directions, and Coulomb friction in tangent direction, as shown in figure 1. By this, MD (Molecular Dynamics, that is a DEM) can be used [10–12] to integrate the motion equations of the grains directly from Newtons' laws. By choosing an appropriated time step, several different methods can be used to integrate these equations.

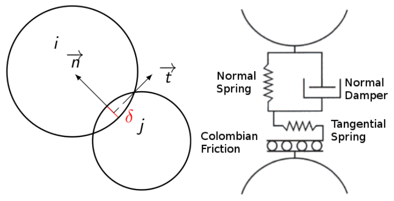

Figure 1. Force model between grain's interactions. i and j are particles that have contact each other. δ is the interpenetration value between the contact ij. **n** is the direction of the normal contact force. **t** is the direction of the tangential friction force.

In MD, a predictor-corrector method is used, and its scheme can be seen in Figure 2. The routine *Predict states* solves the kinematic equations over all grains over a single step. The routine *Detect contacts* find grains in contacts each other and store into a list. For the implementation without routine *Set list of neighbors*, *Detect contacts* look for all possible pair of grains for contacts, at every time step, and requires two loops over all grains. Otherwise, *Detect contacts* focus the search to the list of neighbor grains, and require one loop over the neighbor list. Both cases, it is produced a list of contacts, with the predicted states. The routine *Calculate forces* calculate the contact forces and also gravitational, requiring one loop over all particles and one loop over the list of contacts. The routine *Correct predicted states* corrects the predicted states and requires one loop over all particles. If the routine *Set list of neighbors* have Verlet's list implemented, it will search all possible pairs of neighboring grains, and requires two loops over all grains with a given frequency. If the routine *Set list of neighbors* have Verlet's cells implemented, it will search all neighbors grains' dividing the space in regions and look for possobli pair of contact grains only inside a cell and its neighboring cells. It requires one loop over all grains and other over all grains in the cells'neighbors.

To parallelize the simulation, all barriers should be identified to do a proper implementation. The first barrier is a temporal barrier, and happens in simulation each step. This first barrier can't be jumped by the temporal

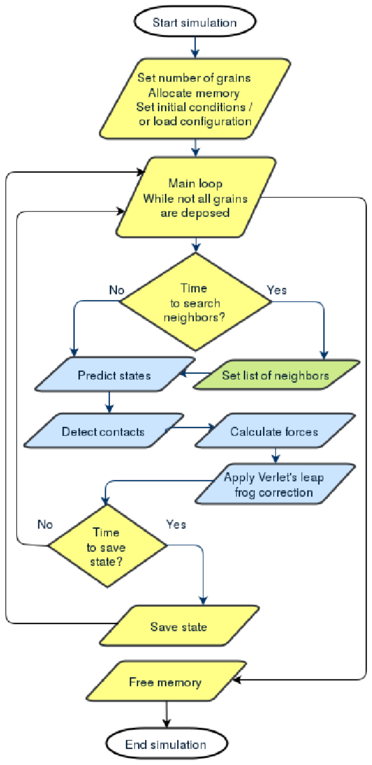

Figure 2. Flowchart of the simulation to simulate granular system using MD. Yellow operations can't be parallelized. Blue operations can be parallelized using CPUs. Green operation is the function that consume most processing, and can be parallelized based onto the type of implementation: *Set list of neighbors* as Verlet's list is the first improvement to reduce time in simulations. *Set list of neighbors* as Verlet's cells is the second improvement to reduce time in simulations.

dependency of the system, as future depends on all past calculation done. Each particle can be treated independently, and at that point, the following routines *Predict states*, *Detect contacts*, *Correct predicted states* and *Set list of neighbors*, using Verlet's list, have no barrier to parallelize, so all loops were done to distribute all calculations to each processor available independently to read and write on memory. In the routine *Set list of neighbors*, using none of Verlet's improvement, the *Set list of neighbors* have no function, once all time steps the contact is calculated inside *Detect contacts*, and no barrier is present in parallelization. If the routine *Set list of neighbors* is using Verlet's cells, one barrier happens to insert grains in the cell it belongs, because to write on one same cell, two processors may try to do at same time. In the routine *Calculate forces* a barrier may happen, depending on implementation. If the 3^{rd} Newton's law is applied on both bodies at same computational sequence when one contact happen, a barrier should be present to prevent memory concurrency by two or more contacts in a particle. This implementation gives an advantage of computation in serial execution that reduces the calculation by half, but a disadvantage in parallel execution by one barrier. The way done here is to calculate the pair of action and reaction separately, in each particle's loop.

The most expansive routine is detect contacts that searches all grains in contact each other, and its computational complexity can be written in function of the number of grains (n), as $O(n^2)$ with Verlet's list, and $O(n \log n)$ with Verlet's cells. Other routines have computational complexity $O(n)$. So, the total complexity of this algorithm can be written as $O(n^2)$ for Verlet's list, and $O(n \log n)$ for Verlet's cells.

2.1 Time spend

The clock time to simulate the system depends on the number of particles, the number of steps the system is simulated, the dynamic of the grains, the number of process running on the machine or cluster, the machine or cluster itself, the method used to implement the solution and many other controllable and uncontrollable variables.

To have an optimized parallel code, one need to know how much time serial functions costs, and their parallelizable portion. The results of time spent by each routine, in seconds, can be found on table 1, table 2 and table 3. The function *Detect contact* is the one which displays best results for application of parallization algorithm. In fact, Verlet's list and Verlet's cells are optimizations for the search algorithm. Both saves computational time and store closer grains into one list. The system do not update the neighbor list all steps of simulation, but only at a given frequency.

Table 1 shown the profile of serial runs of the code with 10^3 grains. For this number of grains, Verlet's cells and Verlet's list have little impact on execution times, but the simple search is the slowest, as expected. Table 2 have the profile of serial running of the code for 10^4 grains, and at this number of grains, Verlet's cells are faster than Verlet's list. Table 3 have the profile of serial running of the code for 10^5 grains, only for Verlet's list and Verlet's cells, because the simplest implementation got no result running after one week with this number of grains, and Verlet's cells are still faster than Verlet's list.

To compare and understand the parallelization, some metrics are defined. One of them is the performance of the system, that is the clock time measured by running that code. Faster codes gives lower results. Our results of performance can be found in figure 3, and for simulations with number of grains higher than 10^3, Verlet's cells gives best results.

The speedup is the comparison between clock time of execution in serial and the clock time of execution in parallel of same algorithm. It gives the tendency of the parallelization of the code. Best results gives higher curves. Our results of speedup, shown in figure 4, evince that parallelization decreased clock time to perform simulations aboce than 10^3 particles in the system. We can also conclude that *Detect contacts* without Verlet's implementations and Verlet's list are much more parallelizable than Verlet's list.

The efficiency is the comparison between speedup and the number of processors used. It gives the average utilization of each processor in the algorithm. Best results tends

Table 1. Clock time spent, in seconds, on each routine of serial running of the code for 10^3 grains for 10^3 time steps.

Function	Simplest	List	Cells
Predict states	0.01	0.03	0.03
Detect contact	22.47	0.15	0.18
Calculate forces	0.01	0.01	0.01
Correct states	0.01	0.04	0.02
List of neighbors	- -	0.22	0.01

Table 2. Clock time spent, in seconds, on each routine of serial running of the code for 10^4 grains for 10^3 time steps.

Function	Simplest	List	Cells
Predict states	0.27	0.19	0.21
Detect contact	2048	1.98	1.84
Calculate forces	0.09	0.06	0.05
Correct states	0.27	0.34	0.30
List of neighbors	- -	23.18	0.05

Table 3. Clock time spent, in seconds, on each routine of serial running of the code for 10^5 grains for 10^3 time steps.

Function	Simplest	List	Cells
Predict states	- -	2.93	2.87
Detect contact	- -	19.87	19.77
Calculate forces	- -	0.93	0.86
Correct states	- -	3.87	3.73
List of neighbors	- -	2,322.96	0.59

to have value 1, the full use of each processor. Figure 4 shown us that *Detect contacts* without Verlet's implementations and Verlet's list uses more the computational resources at same time than Verlet's cells implementation. For higher number of grains, they occupy more the computational resources, justifying the good use of parallelization with large systems.

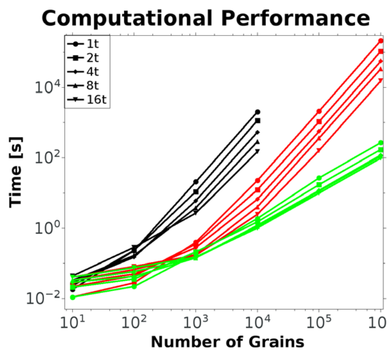

Figure 3. Performance of the system. This compares the results without Verlet's methods (in black), Verlet's list (in red) and Verlet's cells (in green). The number of processors used to simulate varies from 1 to 16, indicated on the legend.

3 Brazil Nut Effect (BNE)

The system has been validated comparing results with the theory of BNE [8, 9, 13, 14]. BNE is a segregation phe-

DOI: 10.1051/epjconf/201714015024

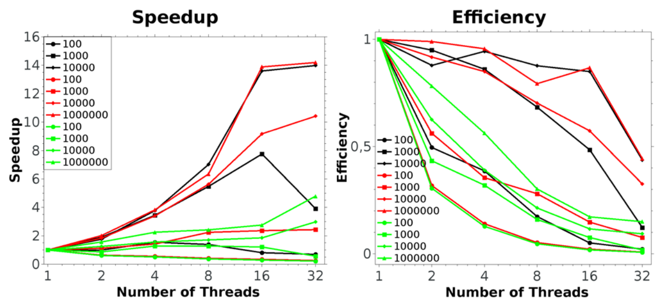

Figure 4. Speedup and Efficiency of the system. This compares the results without Verlet's methods (in black), Verlet's list (in red) and Verlet's cells (in green). The number of grains used to simulate varies from 100 to 1000000, indicated on the legend.

nomena which occurs when a system is shaken and a grain which is larger than the other grains in the media, rises to the surface. Friction is also an important influence to BNE, as can be seen in Figure 5, the larger grain rises in both cases for dimensionless accelerations higher than 1.00, but comparing friction walls with frictionless walls, the intruder with friction walls rises much more than the one without friction.

The parameters of the simulated system were: 2500 grains, uniform polydispersion of the radius of grains around 5%, friction coefficient of 0.5 between grains, and between grains and walls, normal spring stiffness of 1000 adimensional unities [3], tangential spring stiffness of 750 a.u., the radius of the intruder equals 3 times the radius of the larger grains in the media, density of the system equals to π^{-1} in a. u., critical damping coefficient in normal direction, time step of one tenth of the typical colision time for the smallest grains in the system. An imposed sinusoidal force is submitted to the box at 2500 a.u. and the dimensionless acceleration, $\Gamma = A\omega^2/g$, is varying between 0.85 to 1.5.

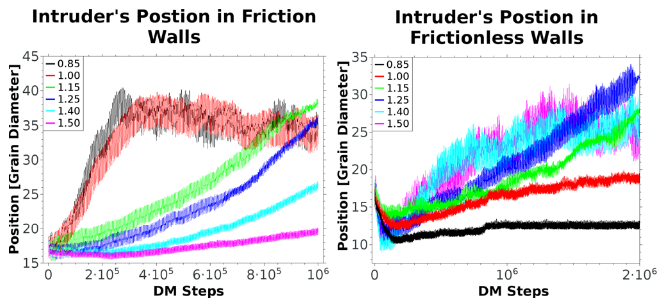

Figure 5. BNE with different amplitudes on the vibration. At left, grains have friction with walls. At right, walls have no friction.

4 Conclusions

We have presented the performance of serial and parallel MD code to simulate granular systems, with 3 different implementations on the computational function that costs most. The implementation that shown best performance was Verlet's cells, and parallelization present higher gain with increasing number of grains. The BNE simulations evince that intruder's ascension is faster and get higher stationary positions when friction is present at grains and walls.

Acknowledgements

We thanks to CEFET-MG that gives us materials and conditions to develop our research. Special thanks to CEFET-MG, FAPEMIG, CAPES and CNPq for funding the research.

References

[1] Radeke, Charles A. and Glasser, Benjamin J. and Khinast, Johannes G., Chemical Engineering Science **65**, 6435-6442 (2010).

[2] Cruz-Hidalgo, Raúl and Kanzaki, T. and Alonso-Marroquin, F. and Luding, S., Powders and Grains 2013 AIP Conference Proceedings **1542**, 169-172 (2013).

[3] Atman, A. P. F. and Claudin, P. and Combe, G., Computer Physics Communications **180**, 612–615 (2009).

[4] Andreotti, B. and Forterre Y. and Pouliquen O., *Granular Media: Between Fluid and Solid* (Cambridge University Press, São Paulo, 2013).

[5] Atman, A. P. F. and Claudin, P. and Combe, G. and Martins, G. H. B., Granular Matter **16** 193-201 (2014).

[6] J. Duran, A. Reisinger, and P. de Gennes, *Sands, Powders, and Grains: An Introduction to the Physics of Granular Materials* (Springer, New York, 1999).

[7] Atman, A. P. F. and Kolb, E. and Combe, G. and Paiva, H. A. and Martins, G. H. B., Powders and Grains 2013 AIP Conference Proceedings **1542**, 381-384 (2013).

[8] Mehta, A., *Granular Physics* (Cambridge University Press, São Paulo, 2007).

[9] Herrmann, H. and Hovi J. and Luding S., *Physics of Dry Granular Media* (Springer, Netherlands, 2013).

[10] Pöschel, T. and Schwager, T., *Computational Granular Dynamics: Models and Algorithms* (Springer, Berlin Heidelberg, 2005).

[11] Allen, M. P. and Tildesley, D. J., *Computer Simulation of Liquids* (Clarendon Press, Oxford, 1991).

[12] Hedman, Fredrik, *Algorithms for Molecular Dynamics Simulations* (Institutionen för fysikalisk kemi, oorganisk kemi och strukturkemi, 2006).

[13] Gutiérrez, G. and Pozo, O. and Reyes, L. I. and Drake, J. F. and Ott, E. and others, Europhysics Letters **67**, 369-372 (2004).

[14] Brito, R. and Soto, R., The European Physical Journal Special Topics **179**, 207-219 (2009).