Powder Technology 324 (2018) 18–35

Contents lists available at ScienceDirect

Powder Technology

journal homepage: www.elsevier.com/locate/powtec

# Grains3D, a flexible DEM approach for particles of arbitrary convex shape - Part II: Parallel implementation and scalable performance

Andriarimina Daniel Rakotonirina[a], Anthony Wachs[b,c,*]

[a] IFP Energies nouvelles, Fluid Mechanics Department, Rond-point de l'Echangeur de Solaize, BP 3, Solaize 69360, France
[b] Department of Mathematics, University of British Columbia, 1984 Mathematics Road, Vancouver, BC V6T 1Z2, Canada
[c] Departement of Chemical and Biological Engineering, University of British Columbia, 2360 East Mall, Vancouver, BC V6T 1Z3, Canada

ARTICLE INFO

*Article history:*
Received 11 September 2016
Received in revised form 22 September 2017
Accepted 19 October 2017
Available online 24 October 2017

*Keywords:*
Granular flow
Discrete Element Method
Angular particles
Parallel computing

ABSTRACT

In [1] we suggested an original Discrete Element Method that offers the capability to consider non-spherical particles of arbitrary convex shape. We elaborated on the foundations of our numerical method and validated it on assorted test cases. However, the implementation was serial and impeded to examine large systems. Here we extend our method to parallel computing using a classical domain decomposition approach and inter-domain MPI communication. The code is implemented in C++ for multi-CPU architecture. Although object-oriented C++ offers high-level programming concepts that enhance the versatility required to treat multi-shape and multi-size granular systems, particular care has to be devoted to memory management on multi-core architecture to achieve reasonable computing efficiency. The parallel performance of our code Grains3D is assessed on various granular flow configurations comprising both spherical and angular particles. We show that our parallel granular solver is able to compute systems with up to a few hundreds of millions of particles. This opens up new perspectives in the study of granular material dynamics.



## 1. Introduction

Discrete Element Method (DEM) based simulations are a very powerful tool to simulate the flow of a granular media. The foundations of the method were introduced by Cundall and Strack [2] in the late seventies. Originally developed for contacts between spherical particles, the method was later extended to polyhedra by Cundall in 1988 [3]. The conceptual simplicity combined with a high degree of efficiency has rendered DEM very popular. However, there are essentially still two bottlenecks in DEM simulations: (i) the non-sphericity of most real life particles and (ii) the generally large number of particles involved even in a small system.

In [1] we addressed issue (i), i.e., the non-sphericity of particles by reviewing the various existing techniques to detect collisions between two non-spherical particles and by suggesting our own collision detection strategy that enables one to consider any convex shape and any size. Issue (ii) can be tackled in two different and complementary ways. The former involves improving the computational speed of classical serial implementations of DEM. This can be achieved by a higher quality programming and smarter algorithms, but there is admittedly a limit in that direction, even with the most advanced implementations. The latter involves dividing the work load between different computing units and hence using distributed computing. Nowadays, there are two (potentially complementary) technologies for DEM distributed computing: CPU [4–9] vs GPU [9–16]. Both technologies have assets and drawbacks. Interestingly, the definition of large-scale computing fluctuates quite a lot from one publication to another publication as well as changes with time and fast-evolving supercomputing resources. While GPU is parallel in essence (multi-threaded), fast on-chip memory is limited in size and global memory access is very slow, which can result in a weak performance of the code [11]. Besides, the built-in parallelism of GPU is not yet fully designed for multi-GPU computations, which may limit the overall performance to that of a single GPU, in particular in terms of system size, i.e., number of particles. However, recent developments have shown that reasonable scalability can be achieved with single- and multi-GPU computations, as summarized in Table 1. Please note that, as emphasized by Shigeto and Sakai [9], the speed ratio 1 GPU/1 CPU in Table 1 and in general one-to-one GPU vs CPU comparisons might not always be fair.

CPU-based DEM codes generally exhibit no limit in number of communicating CPUs (cores) and hence no limit in number of

* Corresponding author at: Department of Mathematics, University of British Columbia, 1984 Mathematics Road, Vancouver, BC V6T 1Z2, Canada.
*E-mail address:* wachs@math.ubc.ca (A. Wachs).

https://doi.org/10.1016/j.powtec.2017.10.033
0032-5910/

**Table 1**
Summary of recent contributions to DEM computations on GPUs. "nr" stands for "not reported".

| Authors | Max number of GPUs | Max number of particles | Speed ratio 1 GPU/1 CPU |
|---|---|---|---|
| Xu et al. [16] | 270 | 10,000,000 | nr |
| Washizawa and Nakahara [13] | 1 | 131,072 | 6 |
| Shigeto and Sakai [9] | 1 | 1,280,000 | 0.87 – 3.4 |
| Tsuzuki and Aoki [12] | 512 | 129,000,000 | nr |
| Gan et al. [15] | 34 | 10,000,000 | 10 |

particles, provided the scalability is maintained at a reasonable level. Communications between cores is achieved using MPI [17]. Although computations with up to a few tens of millions of particles are emerging with GPU-based implementations [10-12,15], simulations with up to a few billions of particles can be envisioned with CPU-based implementations, provided computational practitioners have access to large supercomputers with many thousands of cores [6–8]. The forthcoming new GPU technology is likely to offer similar parallel computing capabilities as the CPU technology, either by improving inter-GPU communications without using CPUs or by speeding up data exchange between GPUs and CPUs. At the time we write this article, this enhanced GPU technology is not available yet. Multi-CPU implementations already have or will soon have to address other challenges related to the evolution from multi-core to many-core technology, i.e., computing nodes have more CPUs that each have more cores. The current multi-core technology also poses tough challenges in terms of memory access and management (that we partly address in this work) but the next generation many-core technology will render these challenges even more crucial. One option to address them involves developing hybrid shared/distributed computing models, i.e., shared on a node with OpenMP and distributed among nodes using MPI [18]. These hybrid implementations might have been optional so far with 8, 16 or even 24 cores per node only, but may become mandatory in the future as the number of cores per node is likely to keep on increasing. Another option is to rethink the programming paradigm, both for many-CPU [19] and many-GPU [20] computing. This is an ongoing effort in the scientific computing community.

DEM is used both in dry granular flow computations and in particle-laden flow computations. Collision detection and resolution is generally the most time-consuming part of a DEM computation. When particles are immersed in a fluid, hydrodynamic interactions reduce the number of collisions between particles and hence computations (corresponding to the DEM solver only, not the solution of the fluid mass and momentum conservation equations) for the same number of particles are faster or conversely a higher number of particles can be considered for the same computing time. Fluidized bed simulations are a typical case of a relatively low number of collisions with respect to the number of particles in the system once the bed is sufficiently fluidized, e.g., the inlet velocity is at least 2–3 times the minimum fluidization velocity. In a simple configuration of fluidization in a box, Pepiot and Desjardins [21] perform computations with up to 382 million of spheres on 4096 cores with a parallel efficiency of 85%. In the field of particle-laden flow simulations, let us mention the long-term effort of the National Energy Technology Laboratory in the development of the open source code MFIX. In the past few years, MFIX, which has historically been known for its Two-Fluid model (MFIX-TFM) and Particle in Cell model (MFIX-PIC), has been enriched by an Eulerian/Lagrangian model (also called DEM-CFD) that relies on a DEM solver for the Lagrangian tracking of particles with collisions. The performance of the parallel DEM solver involved in the so-called MFIX-DEM model is analyzed by Gopalakrishnan and Tafti in [22]. Computations with up to 10 million of spheres in a fully 3D fluidized bed configuration on up to 256 cores with a reasonably satisfactory parallel efficiency are presented. Using the MFIX-DEM model, Liu and Hrenya [23] also investigate its parallel performance in pseudo-2D rectangular fluidized beds. Computations with up to 10 million of spheres on up to about 80 cores show a satisfactory scalability provided the number of particles per core is $10^5$. Liu and Hrenya also address the important question of the physical time that can be computed versus the number of particles that can be computed and show that the balance is controlled by the domain size to particle size ratio, as smaller particles require smaller time steps to resolve collisions. MFIX possesses an active and large community of users and the literature comprises numerous works using MFIX to examine particle-laden flows and in particular fluidized bed configurations. Among many others, let us mention the recent work of Yang et al. [24,25] using MFIX-DEM to study the flow dynamics in a double slot-rectangular spouted bed that contains around 2.6 million of spheres. Finally, Gel et al. [26] recently attempted to improve the parallel performance of MFIX by partly refactoring the code at a rather deep programming level to better fit modern high-performance computing architectures. Significant computing time reductions, up to 8 times improvement, are obtained. This is in line with our effort, presented later in this work, in refactoring our own code Grains3D to attain a satisfactory parallel performance.

Our goal in this paper is to elaborate on a simple domain decomposition based parallel extension of our granular code Grains3D and to assess its computing performance on systems of up to a few hundreds of millions of particles. Please note that most references given above considered spheres or spheroids. The main strength of our implementation is the ability to combine our simple but efficient parallel implementation to our collision detection strategy for non-spherical and angular particle shapes [1], and hence to target large-scale DEM computations of many millions of particles of, e.g., polyhedral shape. In Section 2, we quickly recall the features of our numerical model as already explained in [1]. We then present our parallel strategy in Section 3. In Section 4 we measure the computing performance of our parallel implementation in various granular flow configurations (particle shape, particle load by core, weak scalability). Finally, we discuss parallel computing performances exhibited by Grains3D in Section 5 and highlight the remaining intrinsic limitations of Grains3D and how to relax them.

## 2. Numerical model

The motion of the granular material is determined by applying Newton's second law to each particle $i \in < 0, N-1 >$, where $N$ is the total number of particles. The rigid body motion assumption leads to the decomposition of the velocity vector $\boldsymbol{v}$ as $\boldsymbol{v} = \boldsymbol{U} + \boldsymbol{\omega} \wedge \boldsymbol{R}$, where $\boldsymbol{U}$, $\boldsymbol{\omega}$ and $\boldsymbol{R}$ denote the translational velocity vector of the center of mass, the angular velocity vector of the center of mass and the position vector with respect to the center of mass, respectively. The complete set of equations to be considered is the following one:

$$M_i \frac{d\boldsymbol{U}_i}{dt} = \boldsymbol{F}_i \tag{1}$$

$$\boldsymbol{J}_i \frac{d\boldsymbol{\omega}_i}{dt} + \boldsymbol{\omega}_i \wedge \boldsymbol{J}_i \boldsymbol{\omega}_i = \boldsymbol{M}_i \tag{2}$$

$$\frac{d\boldsymbol{x}_i}{dt} = \boldsymbol{U}_i \tag{3}$$

$$\frac{d\boldsymbol{\theta}_i}{dt} = \boldsymbol{\omega}_i \tag{4}$$

where $M_i$, $\boldsymbol{J}_i$, $\boldsymbol{x}_i$ and $\boldsymbol{\theta}_i$ stand for the mass, inertia tensor, center of mass position and angular position of particle $i$. $\boldsymbol{F}_i$ and $\boldsymbol{M}_i$ are the sum of all forces and torques applied on particle $i$, respectively,