# On the Use of Graphics Processing Units (GPUs) for Molecular Dynamics Simulation of Spherical Particles

R.C. Hidalgo*, T. Kanzaki†, F. Alonso-Marroquin** and S. Luding‡

*Department of Physics and Applied Mathematics, University of Navarra, Pamplona, Navarra, Spain
†Gilogiq Internet Services S.L. Girona, Spain
**School of Civil Engineering, The University of Sydney, Sydney NSW 2006, Australia
‡Multi Scale Mechanics, CTW, UTwente, P. O. Box 217, 7500 AE Enschede, Netherlands

**Abstract.** General-purpose computation on Graphics Processing Units (GPU) on personal computers has recently become an attractive alternative to parallel computing on clusters and supercomputers. We present the GPU-implementation of an accurate molecular dynamics algorithm for a system of spheres. The new hybrid CPU-GPU implementation takes into account all the degrees of freedom, including the quaternion representation of 3D rotations. For additional versatility, the contact interaction between particles is defined using a force law of enhanced generality, which accounts for the elastic and dissipative interactions, and the hard-sphere interaction parameters are translated to the soft-sphere parameter set. We prove that the algorithm complies with the statistical mechanical laws by examining the homogeneous cooling of a granular gas with rotation. The results are in excellent agreement with well established mean-field theories for low-density hard sphere systems. This GPU technique dramatically reduces user waiting time, compared with a traditional CPU implementation.

**Keywords:** MD, DEM, Homogeneous Cooling, Friction and Rotations, Numerical Methods, GPU
**PACS:** 81.05.Rm, 83.10.Rs

## INTRODUCTION

In the last years, rapid advances in computer simulations have led to many new developments in modeling particulate systems. Molecular dynamics simulation (MD) is widely accepted as an effective method in addressing physical and engineering problems concerning dense granular media [1]. The main disadvantages of molecular dynamics algorithms implemented on central processing units (CPU) are the maximum number of particles and the expensive computing time of the simulation.

Graphics processing units (GPU) are designed to rapidly manipulate and alter memory. Their highly parallel structure makes them more effective than general-purpose CPUs for algorithms where processing of large blocks of data is done in parallel. Thus, general-purpose computation on (desktop) graphics hardware (GPGPU) [2, 3, 4] has become a serious alternative for parallel computing on clusters or supercomputers.

Compute Unified Device Architecture (CUDA) is a parallel computing platform, which has been recently introduced by NVIDIA [2, 3]. This platform notably improves the computing performance by exploiting the power of the GPUs. The open source NVIDIA GPU Computing package provides several code samples that help to get started on the path of writing software with CUDA C/C++, OpenCL or DirectCompute. Specifically, the example *particles-CUDA* is a simple algorithm, which includes discrete elements that move and collide within an uniform grid data structure. However, this implementation is by no means optimal and there are many possible further optimizations to this algorithm.

## MD IMPLEMENTATION OF SPHERES ON GPUS INCLUDING ROTATION

We have developed a new hybrid CPU-GPU Discrete Element based on the CUDA-particles example. The first step was to replace the Euler's integrator by a Velocity Verlet integration method. This modification notably improved the numerical output of the algorithm and it guaranties that the total mechanical energy of the system always oscillates around a constant value that corresponds to the exactly solved system. The same goes for other conservative quantities like linear or angular momentum that are, at least nearly, preserved using this *symplectic integrator*. The original collision rule was replaced by a generalized contact law that is more realistic than the linear spring contact. Rotational degrees of freedoms were included and are activatd by a tangential force that depends on history. Accordingly, a neighbor list and a contact list have been also implemented.

The application developed, as most of the GPGPU software, has an heterogeneous architecture. This means that some pieces of code run in the CPU and others in the GPU. The flowchart of the MD method is presented in Figure 1. The first steps of the program consist in the

*Powders and Grains 2013*
AIP Conf. Proc. 1542, 169-172 (2013); doi: 10.1063/1.4811894

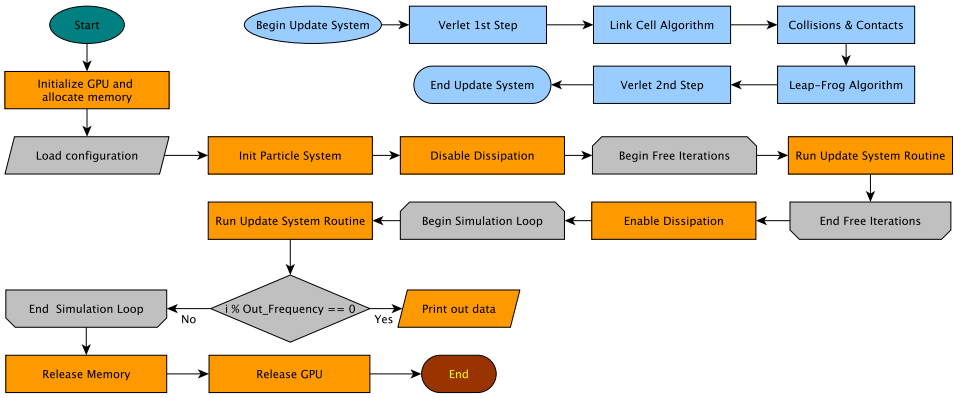


**FIGURE 1.** Flowchart of the granular gas simulation. Operations in gray run on the CPU, subroutines in blue run on the GPU and the ones in oranges run partially in CPU and GPU, and, in most cases, they require data-interchange between CPU and GPU.

initialization of the CUDA-enabled device, the allocation of the necessary memory –in both CPU and GPU– and loading configuration parameters of the granular gas. Initially, the particles are homogeneously distributed in the simulation space with a random velocity for translational and rotational degrees of freedom (this is done on the host and then the particles' information is sent to the GPU device). With the goal to avoid effects of the initial configuration, the dissipation due to particle-particle interaction is disabled, and a number of iterations whith very low dissipation is performed. After that, the energy loss is enabled again and the main loop of the program starts, calling in each iteration the *Update System* subroutine, and with a periodic (lower) frequency printing out the particles information. When the simulation finishes the resources reserved are released and the program ends. In the *Update System* routine is where the MD processes occur. Initially, following the Velocity Verlet integration method, the particles' velocity in the mid-point is calculated and with it the positions are updated. Then, with the aim of minimize the time used by the collisions method, the list with the particles that are neighbors to each other is refreshed. Next, the collisions between particles are computed, calculating the forces and torques that each particle experiences and the list of contacts is updated. Finally the last step of the Verlet and the leap-frog integrator are performed.

As we mentioned above, to define the normal interaction $F_{ij}^N$, we use a linear elastic force, depending on the overlap distance $\delta$ of the particles. To introduce dissipation, a velocity dependent viscous damping is assumed. Hence, the total normal force reads as $F_{ij}^N = -k^N \cdot \delta - \gamma^N \cdot v_{rel}^N$ where $k^N$ is the spring constant in the normal direction, $\gamma^N$ is the damping coefficient in the normal direction and $v_{rel}^N$ is the normal relative velocity between $i$ and $j$. The tangential force $F_{ij}^T$ also contains an elastic term and a tangential frictional term accounting also for static friction between the grains. Taking into account Coulomb's friction law it reads as $F_{ij}^T = \min\{-k^T \cdot \xi - \gamma^T \cdot |v_{rel}^T|, \mu F_{ij}^N\}$ where $\gamma^T$ is the damping coefficient in tangential direction, $v_{rel}^T$ is the tangential component of the relative contact velocity of the overlapping pair, $\mu$ is the friction coefficient of the particles, $\xi$ represents the elastic elongation of an imaginary spring with spring constant $k^T$ at the contact [5], which increases as $d\xi(t)/dt = v_{rel}^T$ as long as there is a non-sliding overlap between the interacting particles [5, 6, 7]. The tangential spring is chosen to be orthogonal to the normal vector [8]. Finally, we solve Newton's equation of motion for all particles. A quaternion formalism is used to describe the rotation of the particles. Finally, the equations of motion are integrated using a Fincham's leap-frog algorithm (rotational) [9] and a Verlet Velocity algorithm (translational) [10].

*Validation* The numerical accuracy of the algorithm has been validated by comparing our results with a mean field model. Specifically, we have examined the homogeneous cooling of rough and dissipative spherical particles. Luding *et al* [12] and Herbst *et al* [13] have found that translational $T$ and rotational $R$ kinetic energy of granular gas of rough particles, in homogeneous cooling state, is governed by the following system of equations

$$\frac{d}{d\tau}T = -AT^{3/2} + BT^{1/2}R \quad (1)$$

$$\frac{d}{d\tau}R = BT^{3/2} - CT^{1/2}R \quad (2)$$

with the constants $A$, $B$ and $C$, whose values depend on the space dimensionality $D$ and the energy dissipation rates as, $A = \frac{1-e_n^2}{4} + \eta(1-\eta)$, $B = \frac{\eta^2}{q}$ and $C = \frac{\eta}{q}\left(1 - \frac{\eta}{q}\right)$ where $\eta = q(1+e_t)/(2q+2)$ (in 3D $q = 2/5$)

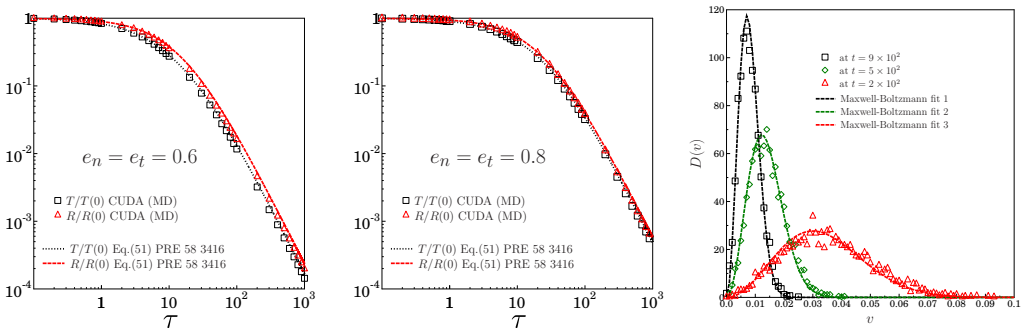


**FIGURE 2.** Evolution of the translational and rotational kinetic energy vs time. a) $e_n = e_t = 0.6$ b) $e_n = e_t = 0.8$. In c) we present the speed distribution obtained for $e_n = e_t = 0.8$ at $t = 2 \times 10^2 s$ (red), $t = 5 \times 10^2 s$ (green) and $t = 9 \times 10^2 s$ (black).

and $e_n$ and $e_t$ are the restitution coefficients on the normal and tangential direction respectively. The mean time between collisions $G = 8(2a)^2 \frac{N}{V} \sqrt{\frac{\pi}{m}} g(2a)$, is used to rescale real time scale accordingly to $\tau = \frac{2}{3} G T^{1/2}(0) t$. The strength of the dissipation can also be included into the characteristic time $\tau = \frac{2}{3}(1 - e_n^2) G T^{1/2}(0) t$ [14]. To compare the numerical output of our code with the theoretical predictions (Equations 2), we have to find equivalent dissipation parameters ($\gamma_n$, $\gamma_t$ and $k_t$) that correspond with specific values of the normal $e_n$ and tangential $e_t$ restitution coefficients. In the simplest approximation, the normal interaction force between two contacting particles is a linear spring $f_{el}^n = k^N \delta$ and a velocity depending viscous damping force $f_{diss}^n = \gamma_n \dot{\delta}$ [11]. Examining the contact evolution one gets a well known differential equation of the damped harmonic oscillator [11]

$$\ddot{\delta} + 2\eta \dot{\delta} + \omega_0^2 \delta = 0 \tag{3}$$

Here $\omega_0 = \sqrt{k/m_{12}}$ is the oscillation frequency of an elastic oscillator and $\eta$ is the effective viscosity, obtained as $\gamma_n = 2\eta m_{12}$ where $m_{12} = m_1 m_2/(m_1 + m_2)$ is the reduced mass. Solving Eq.(3) one can find that the effective restitution coefficient, $e_n = \exp(-\pi\eta/\omega)$ where is the oscillation frequency of the damped oscillator. Combining the equations the following expression is obtained $\gamma_n = \sqrt{(4k_n m_{12})/((\frac{\pi}{\ln \frac{1}{e_n}})^2 + 1)}$.

On the other hand, describing the tangential force between to contacting particles, one can also consider a tangential spring $f_{el}^t = k^t \delta_t$ and a velocity dependent viscous damping $f_{diss}^t = \gamma_t \dot{\delta}_t$ [11]. For simplicity sake here we examined the case $\gamma_t = 0$; for which an analytic expression, relating $k_t$ and $k_n$, can be derived,

$$k_t = \frac{k_n q}{1+q} \left( \frac{\arccos(-e_t)}{\pi} \right)^2 \tag{4}$$

where $q = 2/5$ stands for the $3D$ case [11].

*Numerical Results*. For validation, we have numerically studied the free cooling kinetics of a dilute system of $N = 4096$ spheres confined within a square box with $l = 2m$, resulting in a volume fraction of $V_f = 0.008$. Initially, the particles are homogeneously distributed in the space and their translational and rotational velocities follow Gaussian distributions. To avoid memory effects from the initial conditions, we allow the system to execute several collisions before starting to analyze the particles' temporal evolution. To compare the algorithm performance with the mean field model [12], system of particles with two different restitution coefficient where studied, $e_n = e_t = 0.6$ and $e_n = e_t = 0.8$. The values $k_n = 10^8 N/m$ and a density $\rho = 2000 kg/m^3$ were used. The corresponding dissipative parameters have been calculated using the equations for the normal damping coefficient $\gamma_n$ and for the stiffness of the tangential spring $k_t$. The time step was set to $dt = 10^{-6} s$.

Figure 2 shows the evolution of the translational $T$ and rotational $R$ kinetic energies are. Note that in every case the time scale have been rescaled using the corresponding characteristic time, resulting in $\tau = \frac{2}{3} G T_{tr}^{1/2}(0) t$. As we start from an equilibrium state and the dissipation is low, the system evolves into a homogeneous cooling state. For comparison we also show the corresponding analytic result of Eq. (2) for the same restitution coefficients. The excellent agreement archived for both cases validates the numerical performance of our algorithm.

During the cooling process the velocity statistics was also examined. Low dissipative particles cool down uniformly over a wide range of time. Thus, all the temporal dependences enter through the mean values of the translational and rotational temperature. Such a picture is consistent with the results shown in Fig. 2c. ($e_n = e_t = 0.8$) where the speed distribution, $D(c)$ and ($c = \sqrt{v_x^2 + v_y^2 + v_z^2}$) is presented at several times. In all cases, the speed distribution remains close to a Maxwell-Boltzmann speed distribution $D(c) =$

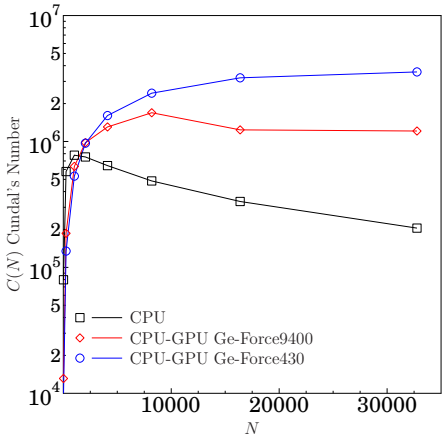


**FIGURE 3.** Cundall number against $N$ for different simulations. In black color a version that run only on the CPU. In red and blue hybrids versions.

$4\pi\left(\frac{1}{\pi v_{mp}^2}\right)^{3/2} c^2 e^{-c^2/v_{mp}^2}$, where $v_{mp}(t)$ is the most probable velocity. For the rotational degrees of freedom we have obtained similar results (data not shown).

*Computational Performance Benchmarks*. In any program that runs in parallel the execution time depends, in conjunction with the hardware, on the number of tasks running simultaneously. We compared the runtime difference between typical MD-algorithms and hybrid CPU-GPU MD-algorithms. A 3D mono-disperse granular gas was used as model example. The first version of the code was a hybrid CPU-GPU algorithm, which uses the GPU to calculate the interaction between particles, whilst the second version fully runs on the CPU. The performance of a particle simulation code is measured by the Cundall number defined as $C = N_T N / T_{CPU}$ where $N_T$ is the number of time steps, $N$ is the number of particles and $T_{CPU}$ is the duration of the simulation in real-time. The CPU version benchmark was performed in a personal computer running Debian GNU/Linux 6.0.2, with a processor Intel® Core™ 2 Quad Q6600 at 2.40 GHz. In the case of the GPU version, the benchmark was performed in the same PC with an NVIDIA® GeForce® GT 430 graphic card, and in an Apple MacBook Pro® with a processor Intel® Core™ 2 Duo at $2.53GHz$ and a graphic card NVIDIA® GeForce® 9400M. In all cases the mean value for 10 different executions –1000000 iterations each one– are presented. The results obtained are shown in Figure 3. Note that when the number of particles is relatively small (from $N = 4$ to $N = 2048$) the Cundall Number is approximately constant, denoting a very similar performance for all cases. As the system size gets larger, however, the performances of the hybrid CPU-GPU algorithm are notably enhanced with respect to the ones only running on the CPU. Furthermore, using and NVIDIA GeForce GT 430 graphic card the simulations executed with the hybrid CPU-GPU algorithm run faster by more than one order of magnitude than on the CPU. It is important to remark, that today there is a new generation of NVIDIA products, which are based on Fermi and Kepler architecture and are optimized for scientific applications. The performance of the hybrid algorithm, on this novel hardware, should be even better.

In summary, we have described the implementation of an accurate molecular dynamics algorithm for monodisperse systems of spheres with rotation, using GPUs. Simulations in GPU are notably faster than traditional CPU methods, the results agree with mean-field theories for low-dense granular systems.

## ACKNOWLEDGMENTS

The Spanish MINECO (Projects FIS2011-26675), the University of Navarra (PIUNA Program) and the University of Sydney Civil Engineering Research Development Scheme (CERDS) have supported this work.

## REFERENCES

1. T. Pöschel and T. Schwager, *Computational Granular Dynamics*. Springer-Verlag Berlin, 2005.
2. J. Owens, D. Luebke, N. Govindaraju, M. Harris, J. Krüger, A. Lefohn, and T. Purcell, *Computer Graphics Forum*, **26**, 80-113, (2007)
3. J. Owens, M. Houston, D. Luebke, S. Green, J. Stone, and J. Phillips, "Gpu computing," *Proceedings of the IEEE*, **96**, 879-899 (2008).
4. Juan-Pierre Longmore, Patrick Marais and Michelle Kuttel *Powder Technology* in Press (2012)
5. P. Cundall and O. Strack, *Geotechnique*, **29** 47-65 (1979)
6. G. Duvaut and J.-L. Lions, *Les Inéquations en Mécanique et en Physique*. Dunod, Paris, 1972.
7. S. Luding *Granular Matter* **10**, 235-246, (2008)
8. T. Weinhart, A. R. Thornton, S. Luding, and O. Bokhove, *Granular Matter* **14**, 531-552 (2012)
9. D. Fincham, "Leapfrog rotational algorithms," *Molecular Simulation*, **8** 165-178 (1992)
10. L. Verlet, *Phys. Rev.*, **165** (201-214 (1968)
11. S. Luding, "Collisions & contacts between two particles," in *Physics of dry granular media - NATO ASI Series E350* (285-304) Kluwer Acad. Publ. (Dordrecht) (1998)
12. S. Luding, M. Huthmann, S. McNamara, and A. Zippelius, *Phys. Rev. E*, **58**, (3416-3425) (1998).
13. O. Herbst, R. Cafiero, A. Zippelius, H. J. Herrmann, and S. Luding, *Physics of Fluids* **17** 107102 (2005)
14. S. Miller and S. Luding, *Phys. Rev. E*, **69** 31305 (2004)