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

Citation: AIP Conf. Proc. 1542, 169 (2013); doi: 10.1063/1.4811894
View online: http://dx.doi.org/10.1063/1.4811894
View Table of Contents: http://proceedings.aip.org/dbt/dbt.jsp?KEY=APCPCS&Volume=1542&Issue=1
Published by the AIP Publishing LLC.
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.

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 guarantees 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 activated 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
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 with 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 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 \( F_{ij}^N = -k^N \cdot \delta - \gamma^N \cdot \dot{v}_{rel}^N \), where \( k^N \) is the spring constant in the normal direction, \( \gamma^N \) is the damping coefficient in the normal direction and \( \dot{v}_{rel}^N \) 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

\[
\frac{d}{d\tau} T = -AT^{3/2} + BT^{1/2}R \\
\frac{d}{d\tau} R = BT^{3/2} - CT^{1/2}R
\]

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

---

**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.
and $e_n$ and $e_t$ are the restitution coefficients on the normal and tangential direction respectively. The mean time between collisions $G = 8(2a)^2 \frac{\gamma}{\sqrt{\pi}}$, is used to rescale real time scale accordingly to $\tau = \frac{1}{2} GT^{1/2}(0)t$. The strength of the dissipation can also be included into the characteristic time $\tau = \frac{1}{2}(1 - e_t^2) GT^{1/2}(0)t$ [14].

To compare the numerical output of our code with the theoretical predictions (Equations 2), we have to find equivalent dissipation parameters ($\gamma_n$, $\gamma$, 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 $f_n^r = k_n \delta$ and a velocity dependent viscous damping force $f_{n\text{diss}} = \gamma_n \dot{\delta}$ [11]. Examining the contact evolution one gets a well known differential equation of the damped harmonic oscillator [11]

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

(3)

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

On the other hand, describing the tangential force between contacting particles, one can also consider a tangential spring $f_t^r = k_t \delta$ and a velocity dependent viscous damping force $f_{t\text{diss}} = \gamma_t \dot{\delta}$ [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,

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

(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 $k_n = 10^3 N/m$ and a density $\rho = 2000kg/m^3$ 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 $dt = 10^{-1}s$.

Figure 2 shows the evolution of the translational $T$ and rotational $K$ kinetic energies. Note that in every case the time scale have been rescaled using the corresponding characteristic time, resulting in $\tau = \frac{1}{2} GT^{1/2}(0)t$. 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, $D(c) = f_c(c) \propto \frac{1}{c^2}$ is presented at several times. In all cases, the speed distribution remains close to a Maxwell-Boltzmann speed distribution $D(c) = f_c(c) \propto \frac{1}{c^2}$.
**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**