EPJ Web of Conferences **140**, 06025 (2017) DOI: 10.1051/epjconf/201714006025
Powders & Grains 2017

BlazeDEM3D-GPU A Large Scale DEM simulation code for GPUs

Nicolin Govender[1,3,4,5,⋆], *Daniel* Wilke[2,⋆⋆], *Patrick* Pizette[3,4,⋆⋆⋆], and *Johannes* Khinast[1]

[1] *Research Center Pharmaceutical Engineering, GmbH, Graz , Austria*
[2] *Department of Mechanical and Aeronautical Engineering, University of Pretoria, South Africa*
[3] *Mines-Douai, LGCgE GCE, F-59508 Douai, France*
[4] *Université de Lille, 59650 Lille, France*
[5] *Department of Mechanical Engineering , University of Johannesburg, South Africa*

Abstract. Accurately predicting the dynamics of particulate materials is of importance to numerous scientific and industrial areas with applications ranging across particle scales from powder flow to ore crushing. Computational discrete element simulations is a viable option to aid in the understanding of particulate dynamics and design of devices such as mixers, silos and ball mills, as laboratory scale tests comes at a significant cost. However, the computational time required to simulate an industrial scale simulation which consists of tens of millions of particles can take months to complete on large CPU clusters, making the Discrete Element Method (DEM) unfeasible for industrial applications. Simulations are therefore typically restricted to tens of thousands of particles with highly detailed particle shapes or a few million of particles with often oversimplified particle shapes. However, a number of applications require accurate representation of the particle shape to capture the macroscopic behaviour of the particulate system. In this paper we give an overview of the recent extensions to the open source GPU based DEM code, BlazeDEM3D-GPU, that can simulate millions of polyhedra and tens of millions of spheres on a desktop computer with a single or multiple GPUs.

1 Introduction

The discrete element method (DEM) has become the *de facto* standard to simulate particulate materials for which the discrete nature of particles and importance of quantifying particle-scale effects cannot be neglected. Consequently, the need for large scale discrete element simulations that can quantify the effect of particle-scale changes on the macro-scale response is becoming increasingly more important in industrial bulk material handling applications, where computational efficient continuum models fail to give reliable predictions [1].

Established discrete element software frameworks rely mostly on multi-core central processing unit (CPUs) computing platforms [2]. This is exemplified by the number of commercial and open-source discrete element codes that include PFC by Itasca , EDEM by DEM Solutions, StarCCM+ by CDAdapco, MecuryDPM by the Univeristy of Twente, BLOCKS3D by the University of Illinois, Y-DEM by Queen Mary University, Millsoft [3] by the University of Utah , LIGGGHTS by Johannes Kepler University in Linz, ELLIPSE3H by New Mexico University, YADE by Grenoble University and SAMADII/DEM by MetaRiver Technology.

Although the GPU (Graphics Processor Unit) is an ideal match for DEM simulations due to the highly threaded nature of the GPU, the technology is fairly new compared to the traditional CPU platform and the associated learning curve for GPU development high. Furthermore only an efficiently implemented GPU DEM solution will yield significant performance benefits over the CPU. Rocky by Granular Dynamics, XPS by RCPE Austria [4], and BLAZE-DEM3D-GPU (opensource) [5] are the only packages capable of efficiently modelling large scale simulations on the GPU. The parallelization benefits of the graphical processing unit (GPU) [6–10] is becoming increasingly more important as an alternative computational platform for discrete element simulations [11]. The GPU seems to be an essential computing platform to scale simulations to millions of particles within in a realistic computing time frame and financial budget. Although the GPU has numerous advantages it also presents various challenges to implementations of the discrete element method. The complexity of resolving contact for non-trivial particle shapes like polyhedra poses particular challenges on the GPU due to the divergent nature of polyhedral contacts. This is a direct result of the limited computational ability and memory restrictions on the GPU and consequently the majority of the effort has been directed to spherical particles on the GPU [2, 9, 10, 12].

⋆ e-mail: govender.nicolin@gmail.com
⋆⋆ e-mail: wilkedn@gmail.com
⋆⋆⋆ e-mail: patrick.pizette@mines-douai.fr

2 BlazeDEM3D-GPU

BlazeDEM3D-GPU that has been demonstrated to simulate (i) tens of millions of mono-disperse spherical particles and (ii) millions of mono-disperse polyhedral particles within a realistic time frame on a desktop computer using a single or multiple GPUs [6][13][14]. BlazeDEM3D-GPU is developed for NVIDIA graphics cards using the available CUDA programming framework [15], allowing for a wide variety of GPU cards to be considered. BlazeDEM3D-GPU [13] has been extensively validated in the simulation of tumbling mills [14] and hoppers [16]. BlazeDEM3D-GPU was designed for the GPU hence the code design is similar to a dynamics engine, so the user is free to build a complex simulation using the various objects provided. This is in contrast to the fixed simulation types offered by commercial codes like Rocky and EDEM, where you can only do a "single" simulation eg ball mills, chutes, hoppers etc as standalone simulations. With BlazeDEM3D-GPU you can simulate a complete process e.g. a belt feeder that feeds into a silo that discharges through a cute into a ball mill. This allows a typical engineer to perform highly specialized simulations.

2.1 Supported Particle Types

The BlazeDEM3D-GPU framework allows for efficient representation and computation for poly-disperse systems for both spherical and polyhedral shaped particles. Numerous particles are currently supported by BlazeDEM3D-GPU that include spheres, convex polyhedra and non-convex polyhedra as depicted in Figure 1(a)-(c). The native supported polyhedral shaped particles include angularity as opposed to popular clumped sphere approaches, as shown in Figure 1(d), that neglect particle angularity. The importance of particle angularity for industrial applications has been investigated a number of times [1, 17].

2.2 World Object Representation

BlazeDEM3D-GPU efficiently represents world objects e.g. explicit representations of planar world objects as depicted in Figure 2.1(c) instead of conventional triangulation or particle representations of world objects as depicted in Figure 2(a) and Figure2(b) [10, 12, 18]. This minimalist representation allows for particle-world contact detection and resolution that is memory and computationally efficient and scalable with particles on the GPU.

While these conventional representations allow for generality in that any surface can be decomposed into particles or triangles it is computationally expensive and requires significant memory. Since we aim to utilize primarily the 48 KB of constant memory to get the most efficient algorithm on the GPU we use a minimalistic surface representation for first order surfaces which requires just the storage of the vertices of the surface. Note that any number of vertices can make up a surface provided they are coplanar. While a single surface can still be represented by

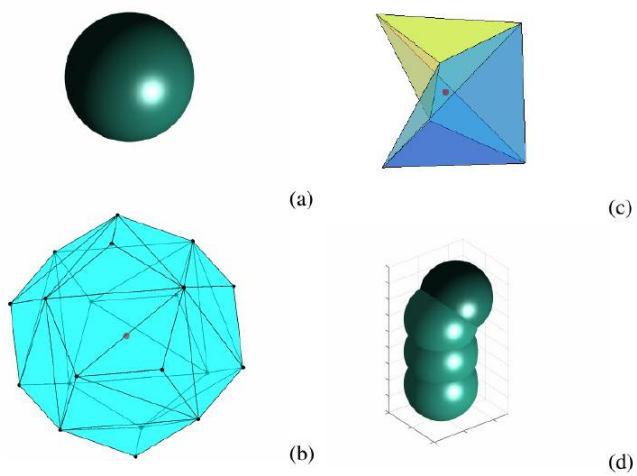

Figure 1. (a) Spherical and (b) native polyhedral shaped particles with angularity (c) native non-convex polyhedral shaped particles are supported by BlazeDEM3D-GPU, as opposed to (d) composite spherical particle approaches that neglect the angularity of particles.

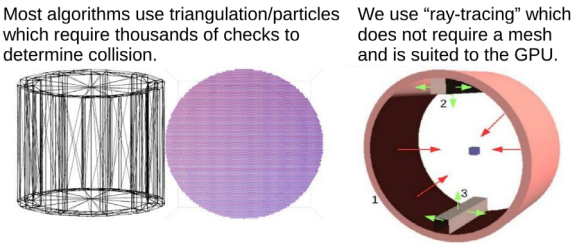

Figure 2. Two world geometry representation strategies.

a collection of smaller surfaces/triangles we use geometric primitives which we term "macro objects" for higher order surfaces such as cylinders which results in significantly reduced memory storage and computational cost. This approach of minimalistic world geometry representation is now also used in the CPU based open-source code MecuryDPM for spherical particles.

The BlazeDEM3D-GPU framework is well documented in [6, 13, 14] and is freely available under the modified BSD-3 license [5].

2.3 Particle-World Contact Detection

The framework considers particle objects that interact with each other and their surroundings described by world objects. The world objects are usually decomposed as volume or surface objects. The method used to determine the contact points between a particle and surface is termed a "ray-tracing approach" [6, 13]. Since convex polyhedra are represented as just a collection of surfaces in BlazeDEM3D-GPU, the same approach is used [13] to determine contact between polyhedra. This approach requires basic algebraic operations and is not iterative like the common-plane approach and hence well suited to the GPU. The details on the particle-world algorithms can be

found in [13, 14], here we just provide a broad overview of the detailed particle-particle interaction.

2.4 Polyhedron Contact Detection

The initial BlazeDEM3D-GPU architecture resolved contact purely based on penetration depth [6, 13]. The contact strain energy therefore varied linearly with penetration depth independent of the volume of material involved during contact. This assumption proved to be not overly restrictive for loosely packed particulate systems but proved unstable for closely packed systems. This simplistic contact model also exhibited inconsistent and sometimes unstable contact resolution when contact was resolved close to the edges and vertices. This sparked the inclusion of a volume based contact model. In 2016, BlazeDEM3D-GPU included a volume-based contact model that exactly resolves the contact volume and contact normals as depicted in Figure 3(a)-(b). The contact volume between two convex particles in contact is also convex. Firstly, all the contact points are established. Thereafter an efficient in-house developed GPU-based convex-hull is used to resolve the vertices and faces of the contact volume. Finally, the contact volume is established and computed, while the contact normals are resolved by analytically integrating the contact surface associated with a particle.

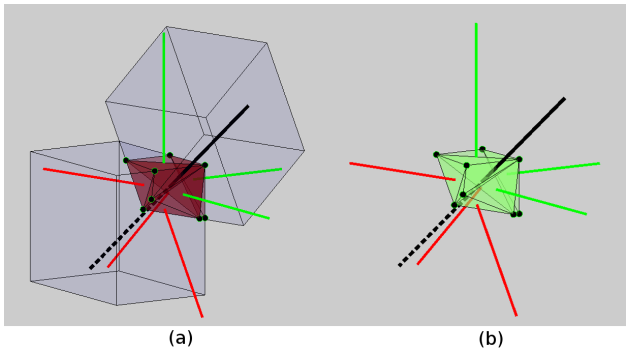

Figure 3. BlazeDEM3D-GPU initially resolved contact based on a penetration depth but recently included a (a) volume-based contact resolution in which the (b) contact volume and normals are exactly resolved enhancing the contact stability of polyhedral shaped particles. Red and green lines indicate face normals and the black solid and dashed lines indicate contact normals.

2.5 Multi-thread GPU collision detection design

Collision detection is computationally the most expensive part of a DEM simulation accounting for as much as 90% of the total simulation time for polyhedral shaped particles. Thus importance is placed on implementing our algorithms so that the take advantage of the parallel nature of the GPU. However regardless of how computational efficiently we implement an algorithm, if we do not utilize memory correctly and minimize transactions then the actual number of threads executing in parallel will be significantly reduced due to register pressure [19]. This is termed the occupancy of a particular method (kernel) executing on the GPU.

To increase the parallel efficiency of collision detection we split the detailed phase into three independent tasks which can be executed concurrently for spheres. Spheres only require a Neighbour Search while polyhedra require the Neighbour Search followed by a detailed particle contact search. Note that typically the particle-particle contact takes up most of the time. We also split the particle-world contact between surfaces and macro objects as different kernels to minimize divergence and reduce the number of registers used by the kernel.

Currently all tasks are run simultaneously on different streams on each GPU.

3 Multi-GPU Simulations

BlazeDEM3D-GPU is able to use multiple GPUs on the same node with an efficiency of 95 % for 2 GPUS and 90 % for 4 GPUS for polyhedra. This scaling is achieved due to our algorithms hiding most of the data transfer cost for polyhedra by overlapping it with compute steps. For spherical particles there a significantly less computational demand thus hiding the data-transfer as efficiently is not possible. While we get an efficiency of 90% for two GPUs the scaling to 3 is poor for less than 10 million particles. It should be noted that the scaling results is for an arbitrary particle distribution, for silos or chutes for example the domain can be decomposed, thus requiring only particles at the interface" to be transferred. In this case scaling is almost linear with the number of GPUs provided the problem size increases to ensure the compute loads are balanced.

4 Particle Shape Simulation Results

The spatial and velocity flow profiles is depicted in Figure 4 for polyhedra and spherical particles for a 0^o hopper configuration. Consider the velocity field at 2.0s, 2.5s and 3.0s for the polyhedra. It is noteworthy that at 2.0 s the effective funnel discharge width is larger than after 3.0 s when the velocity profiles at the orifice are considered. In addition, the flow extends further to the walls up until 2.0s. After 3.0s the flow is more localized. This clearly demonstrates that particle shape needs to be taken into account.

To demonstrate the importance of large scale simulations we simulated a SAG mill with 35% fill using three models. Firstly, as a 2D model using N=6744 particles. The remaining two models were 3D models, a 10% slice of the length of the mill using N=385 534 particles and the full mill using N=4×10^6 particles. The difference in response between the 2D and 3D models is evident resulting in an underestimation of the power draw for the simplified 2D model. We obtained a good match with an error of only 5 % for the power draw for the two 3D models further demonstrating the predictive capability of GPU DEM.

The simulations in this paper were done using the BlazeDEM3D-GPU code developed by the authors on a

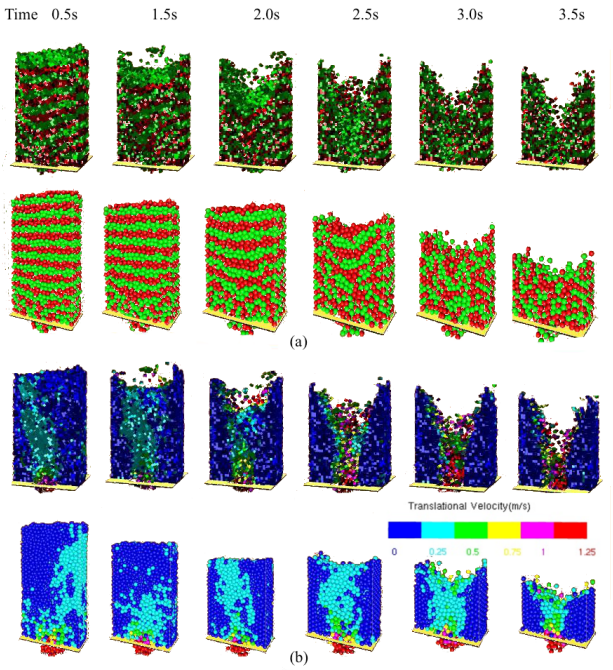

Figure 4. (a) Flow patterns and (b) velocity field for monodisperse cubic shaped and spherical particles discharging from the 0^o hopper configuration at six different time steps.

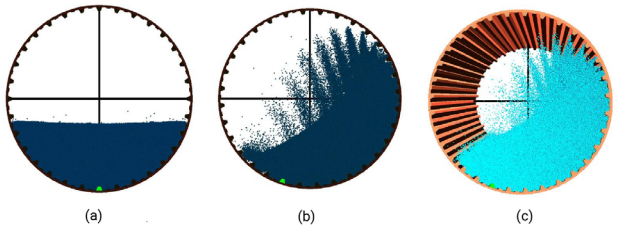

Figure 5. Ball mill steady state simulation with 35% filling modelled as a (a) 2D simulation using N=6744 particles, (b) 10% slice of the length using N=385 534 particles and (c) the full mill using N=4×10^6 particles, all views are orthogonal.

commodity NVIDIA GTX 980Ti graphics card. We required seven hours of wall-time and (900 MB of GPU memory) for a one second simulation time of over a million polyhedral shaped particles in a tightly packed state (before discharge) compared to 18 months using BLOCKS3D and 8.5 days using an impulse-based DEM approach [20] while using a more accurate contact detection resolution. A simulation of over a million spherical particles required just 3 minutes of wall time (300 MB of GPU memory) for one second simulation time.

5 Conclusion

In this paper the effect of particle shape and number on the simulation response was investigated. The need for large scale simulations and the value of our GPU DEM implementation to perform large scale simulations, within realistic time frames, was demonstrated with a 4 million particle simulation.

Acknowledgements

The support of NVIDIA with the donation of GPUs used in this paper is greatly appreciated.

References

[1] Y. Wang, J.Y. Ooi, Procedia Engineering **102**, 765 (2015)
[2] Shigeto, Sakai, Particuology **9**, 389 (2011)
[3] R. Venugopal, R. Rajamani, Powder Technology **115**, 157 (2001)
[4] C. Radeke, B. Glasser, J. Khinast, Chemical Engineering Science **65**, 6435 (2010)
[5] N. Govender, D.N. Wilke, S. Kok, SoftwareX (2016)
[6] N. Govender, D. Wilke, S. Kok, R. Els, JCAM **270**, 63 (2014)
[7] Zhang, et.al., Advances in Engineering Software **60**, 70 (2013)
[8] R. Rajamani, S. Callahan, J. Schreiner, *DEM Simulation of mill charge in 3D via GPU computing* (Proceeding of the SAG conference, Vancouver, 2011)
[9] J. Xu, et.al., Particuology **9**, 446 (2011)
[10] M. Hromnik, *Masters Thesis: A GPGPU implementation of the discrete element method applied to modeling the dynamic particulate environment inside a tumbling mill* (University of Cape Town, www.uct.ac.za, 2013)
[11] J. Gan, Z. Zhou, A. Yu, Powder Technology **301**, 1172 (2016)
[12] J. Longmore, P. Marais, M. Kuttel, Powder Technology **235**, 983 (2013)
[13] N. Govender, D. Wilke, S. Kok, Journal of Applied Mathematics and Computation **http://dx.doi.org/10.1016/j.amc.2014.10.013** (2014)
[14] N. Govender, R. Rajamani, D. Wilke, S. Kok, Journal of minerals engineering **http://dx.doi.org/10.1016/j.mineng.2015.05.010** (2015)
[15] J. Sanders, E. Kandrot, *CUDA by example*, Vol. 12 (2010)
[16] N. Govender, P. Pizette, D.N. Wilke, N. edine Abriak, *Validation of the GPU based BLAZE-DEM framework for hopper discharge*, in *Proceedings of IV International Conference on Particle-Based Methods* (2015)
[17] P.A. Langston, A.J. Matchett, F.Y. Fraige, J. Dodds, Granular Matter **11**, 99 (2009)
[18] H. Abou-Chakra, J. Baxter, U. Tuzun, Advanced Powder Technology **15**, 63 (2004)
[19] NVIDIA, *Cuda 6* (2014), `http://www.nvidia.com/cuda`
[20] S. Jaelee, *Developments in large scale discrete element with polyhedral particles simulations*