Discrete Element Simulation of Mill Charge in 3D using the BLAZE-DEM GPU framework.

Nicolin Govender[1,3] , Raj K Rajamani*[2], Schalk Kok[1], Daniel N Wilke[1]

[1] *University of Pretoria, Department of Mechanical and Aeronautical Engineering, Pretoria, 0001, South Africa*

[2] *Metallurgical Engineering Department, University of Utah 135 South 1460 East Salt Lake City, Utah-84112, USA*

[3] *Advanced Mathematical Modeling CSIR, Pretoria, 0001, South Africa*

Abstract

The Discrete Element Method (DEM) simulation of charge motion in ball, semi autogenous (SAG) and autogenous mills has advanced to a stage where the effects of lifter design, power draft and product size can be evaluated with sufficient accuracy using either two-dimensional (2D) or three-dimensional (3D) codes. While 2D codes may provide a reasonable profile of charge distribution in the mill there is a difference in power estimations as the anisotropic nature within the mill cannot be neglected. Thus 3D codes are preferred as they can provide a more accurate estimation of power draw and charge distribution. While 2D codes complete a typical industrial simulation in the order of hours, 3D codes require computing times in the order of days to weeks on a typical multi-threaded desktop computer. This paper introduces a 3D GPU code based on the BLAZE-DEM framework that utilizes the Graphical Processor Unit (GPU) via the NVIDIA CUDA programming model. Utilizing the parallelism of the GPU a 3D simulation of an industrial mill with four million particles takes 1.16 hours to simulate one second (12 FPS) on a GTX 880 laptop GPU. This new performance level may allow 3D simulations to become a routine task for mill designers and researchers. Furthermore the shorter compute time can elevate the physics included in the computations to a higher level wherein ore particle breakage and slurry transport can be included in the simulation. In this paper we verify our GPU code by comparing charge profiles and power draw obtained using the CPU based code Millsoft and pilot scale experiments. Finally, we show computations for plant scale mills.

1. Introduction

1.1. Background and Motivation

Since the first application of the discrete element method (DEM) for the simulation of grinding mills by Rajamani [1] in 1990 there has been a phenomenal growth in the variety of ways this technique is used in the mining industry. Prior to DEM, Powell's [2] single ball trajectory in rotary mills was a key advancement in understanding lifter relief angle on the trajectory of charge. This approach continues to serve the mining industry even today.

In the late 90s two-dimensional DEM codes were the norm, due to the ease of execution on a personal computer with a single central processing unit (CPU). On the other hand, three-dimensional simulations promise greater accuracy of simulated results at the expense of computing time. At the outset it is useful to discuss the merits of 3D code in comparison with 2D code. The 2D code executes in a matter of hours on a CPU. It has been heavily used in hundreds of mining operations for annual or semiannual replacement of shell lifters [3]. This code has impacted the production, capacity and liner life of ball mills, autogenous mills and semi-autogenous (SAG) mills. The three-dimensional simulations are more accurate because the momentum transfer between balls and rock particles in the axial direction of the mill is accounted for. This opens the door for new insights when utilizing large-scale 3D simulations. 3D simulations has not been readily available to researchers and mill designers since execution times are of the order of weeks on a single CPU for a typical plant size mill. This then becomes impractical to pursue on a routine basis.

Regardless of the severe computational burden of 3D simulations, a number of successes have been reported. In 2001 Venugopal and Rajamani [4] presented a 3D DEM computational framework and compared it against the power draft in a laboratory scale 90 cm diameter mill. In the same year Rajamani and Mishra [5] used the same 3D code for the prediction of power draft in plant scale mills. Herbst and Nordell [6] combined

3D DEM with smoothed particle hydrodynamics (SPH) and the finite element method (FEM) to simulate slurry and solid charge motion, ore particle breakage and liner wear. Cleary[7] demonstrated the sensitivity of charge behavior and power draft of a 5m diameter ball mill to liner geometry and charge composition using a 3D code. There are continued advances in the simulation of breakage and slurry flow incorporating all the details in three-dimensional simulations. Morrison and Cleary [8] describe the evolution of "Virtual Comminution Machine", a simulation code that simulates breakage and slurry transport in tumbling mills. In their simulation both the discrete element method and smoothed particle hydrodynamics are employed for slurry and pebble flow through the grate slots and the pulp lifter. Cleary and Morrison [9] show that 3D DEM combined with SPH is a viable tool for analyzing mineral processing equipment such as mills and twin deck screens. In a more recent study Alatalo et al. [10] compared the experimental deflection of a lifter in a ball mill with 3D predictions made with EDEM [11], a commercial DEM code. They concluded that 3D simulations agree better with predicted experimental values than 2D simulations.

1.2. Computational Aspects

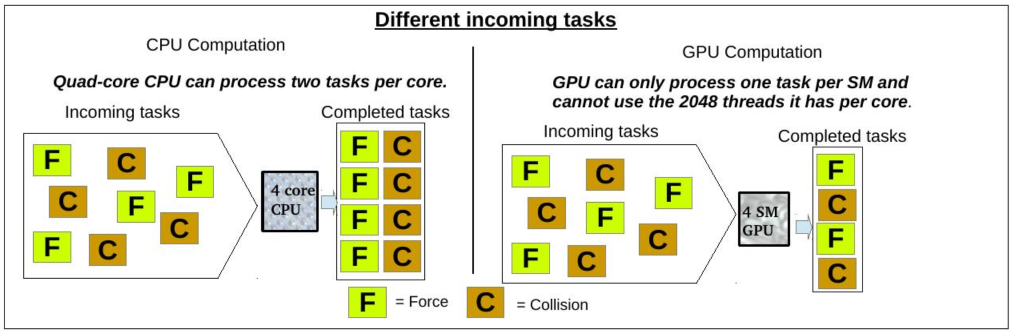
A full 3D simulation of a mill will give valuable insights into the dynamics within a mill which can improve energy efficiency resulting in savings of thousands of dollars. An emerging trend of the past few years is the implementation of scientific and engineering solutions on a new class of processors termed General Purpose Graphical Processor Units (GPGPU) [12, 13, 14], which offers CPU cluster computing performance at a fraction of the cost. Rajamani et al. [15] shows a speed increase of up to 50x over CPU implementations for mill charge motion while Govender et al. [16] showed a speed increase of up to 132x for polyhedral particles.

1.2.1. GPU hardware

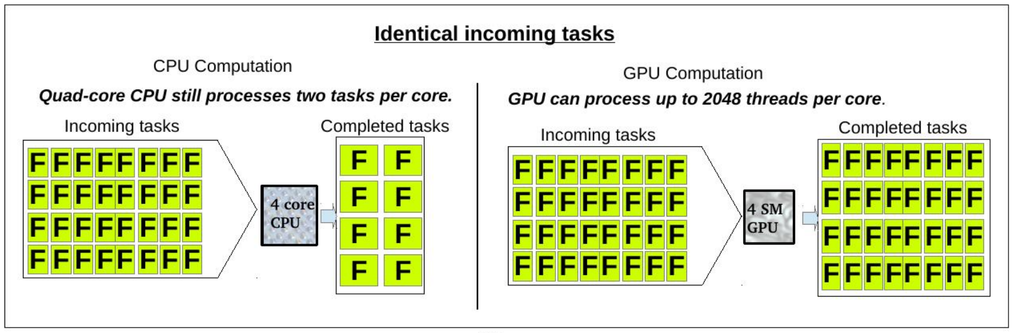
The Graphic Processor Unit (GPU) was initially developed to reduce the computational burden on the CPU during the rendering process which involves the manipulation of millions of pixels on a screen simultaneously. This required that it be made up of mostly Arithmetic Logic Units (ALU) enabling it to perform these arithmetic operations in bulk. This is opposed to control and logic in the case of the CPU [17], which must be able to perform the complex logical operations that is needed to run an operating system. A Streaming Multiprocessor (SM) on a GPU is equivalent to a core on the CPU. Each SM on a Kepler GK110 GPU can launch 2048 threads which are only capable of performing identical tasks (Single Instruction Multiple Data (SIMD))[18]. Each CPU core is capable of launching two threads which can work independently and perform different tasks in each thread at higher clock speeds.

Figure 1 illustrates the type of tasks that each unit excels at in computational performance. The limited

GPU outperforms the versatile CPU in spite of the considerably lower clock rate when processing identical tasks. In order to realize a speed up on the GPU we need to ensure that our DEM algorithm is completely decoupled and expressed as SIMD tasks, in that we carry out the same instructions on different data elements which are particles in the case of tumbling mills.



(a)



(b)

Figure 1: Comparison between CPU and GPU task processing for the case of (a) different incoming tasks and (b) identical incoming tasks.

1.2.2. GPU software platform

The NVIDIA developed CUDA programming model [18] provides access to the GPU from a variety of high level programming languages such as C++, Java and Python. CUDA batches threads into blocks (maximum 1024 threads) for execution on a SM. Threads within blocks can access fast shared memory with each thread in turn having access to its own 32 bit registers (fastest memory available). CUDA allows us to create thousands of thread blocks containing millions of threads which get scheduled for execution on the hardware as SMs become available. Note that we do not have control of the execution order of blocks, so an algorithm must not dependent on the order that threads are executed in. In addition, the execution of a block will only complete once all threads within the block have reached an end point. This is very important and requires us to design algorithms that require similar times to complete for all threads to best utilize the parallelism on the GPU. In this manuscript we show how plant scale mill simulations can be done in a matter of hours and within a day for very large mill simulations.

1.3. Discrete Element Method

The details of discrete element calculations are well documented in literature. Here, we describe the computational steps necessary to implement our algorithm on the GPU. The flow-diagram in Figure 2 describes the DEM process that we model. The bulk of computational time is spent on neighbor searching and collision detection.

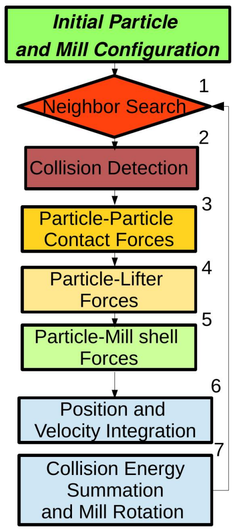

Initial Particle and Mill Configuration
Neighbor Search
1
Collision Detection
2
Particle-Particle Contact Forces
3
Particle-Lifter Forces
4
Particle-Mill shell Forces
5
Position and Velocity Integration
6
Collision Energy Summation and Mill Rotation
7

Figure 2: Flow chart of DEM simulation procedure.

An assumption in many DEM simulations is that particles are considered to be perfectly rigid for the duration of collision contact. In reality perfectly rigid particles do not exist, as all bodies will experience (to some extent) local deformations during contact. These deformations however occur on a time scale which is much smaller than what is required for capturing the macroscopic behavior of a system. Thus it is often sufficient to use a constitutive law, such as a linear spring, to model contact forces. Computing the time evolution of the system requires us to solve simultaneously Newton's equations of motion for all contacting particles, which on current hardware (2014) is only possible for a few thousand rigid bodies [14]. Hence, we assume that there are only binary contacts between particles at any given time. The total force acting on a particle is obtained by summing the individual contributions of all the binary contacts of a particle per time-step. This is a good approximation for many realistic applications that are not densely packed, provided the particles are of a similar size and move very little during a time-step. Hence, computations can be carried out in parallel for all particles independent of each other.

In summary the assumptions that we make are:

1. Rigid bodies with single point contact,
2. Binary contact,
3. Local short-range interactions,
4. Non-incremental friction model and
5. Similar particle size.

These assumptions result in a system that is completely decoupled and can be expressed as a Lagrangian type process in which we are able to simulate the motion of individual particles independently of each other [19] resulting in our algorithm being ideally suited to the GPU.

1.4. Contact Detection

Our mill simulation code is built on the BLAZE-DEM GPU framework developed by Govender et al. [16, 20]. The collision detection algorithm is based on the geometry class classification as described by Govender et al. [16] for the GPU architecture:

1. We firstly do a broad phase check if the particle is beyond the cylinder with radius $\mathbf{r}$ that bounds all the lifters (determined by lifter with largest radius).
2. If a particle passes this check we then loop over all lifters to check if there is intersection between the particle and the bounding cylinder with radius $\mathbf{r}_{\text{lifter}}$ of a lifter, as depicted in Figure 3(a).

Heuristic 1 requires $\mathcal{O}(K)$ computations and heuristic 2 requires $\mathcal{O}(K)$ computations where K is the number of lifters and N is the number of particles.

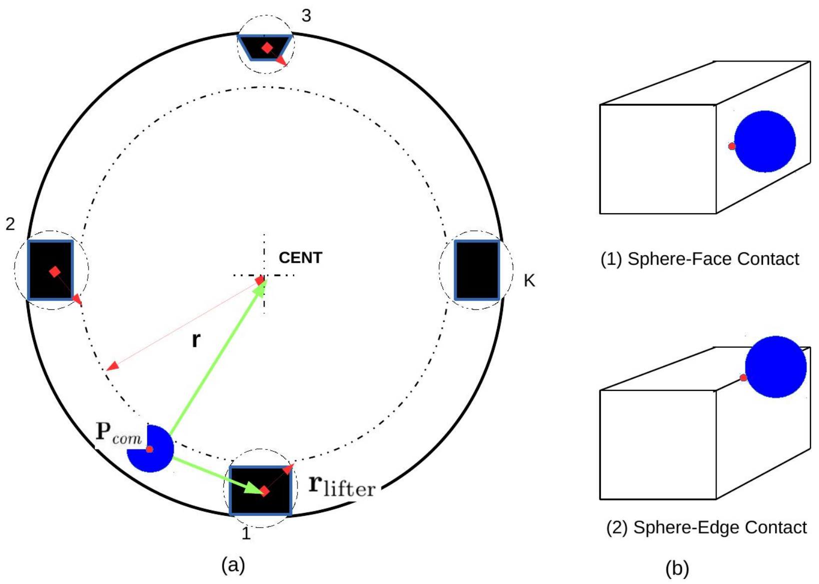


Figure 3: (a) Particle-lifter broad-phase collision detection and (b) detailed collision detection.

If there is an intersection between a ball and a lifter we then find the contact normal and penetration distance as described in Algorithm 1.

Algorithm 1 Particle-lifter detailed collision detection.

```
1. Loop over all lifter faces
   (a) Compute the distance $d = \mathbf{n}_{\text{face}} \cdot (\mathbf{P}_{\text{com}} - \mathbf{C})$ between the lifter face and particle. Here $\mathbf{P}_{\text{com}}$ is the center of mass position of the particle, $\mathbf{C}$ is the center of the face and $\mathbf{n}_{\text{face}}$ the face normal.
   (b) If this distance d is less than the particle radius we have possible contact.
       i. The contact point is given by $\mathbf{P}_{\text{contact}} = \mathbf{P}_{\text{com}} + \text{d}.\mathbf{n}_{\text{face}}$ .
       ii. We now check if the contact point is actually on the face as (a) indicates contact for an infinite plane.
       iii. We check if the penetration $\delta = \text{d} - \text{r}$ is valid (max 5% of radius) ($\delta < 0.05\text{r}$).
       iv. We have contact at point $\mathbf{P}_{\text{contact}}$ with normal $\mathbf{n}_{\text{face}}$ and penetration distance $\delta$.
2. If there is no contact with the faces we do a check for contact with edges, which is computationally more expensive.
3. Loop over all lifter edges
   (a) Compute the vector $\mathbf{L}_{\text{PE}} = (\mathbf{P}_{\text{com}} - \mathbf{E}_{\text{i}}^{0})$ that gives the shortest distance between the particle and lifter edge, where $\mathbf{E}_{\text{i}}^{0}$ is a vertex on the lifter edge.
   (b) We now check if this vector is valid. If $(\mathbf{E}_{\text{i}}^{\text{Dir}} \cdot \mathbf{L}_{\text{PE}}) > 0$, where $\mathbf{E}_{\text{i}}^{\text{Dir}}$ is the direction along the lifter edge, then
       i. We compute the contact point $\mathbf{P}_{\text{contact}} = (\mathbf{E}_{\text{i}}^{0} + \parallel \mathbf{L}_{\text{PE}} \parallel \mathbf{E}_{i}^{\text{Dir}})$ .
       ii. We check if the distance $\text{d} = \parallel \mathbf{P}_{\text{com}} - \mathbf{P}_{\text{contact}} \parallel$ between the point and the particle is less than the radius.
       iii. We check if the penetration $\delta = \text{d} - \text{r}$ is valid (max 5% of radius) ($\delta < 0.05\text{r}$).
       iv. We have contact at point $\mathbf{P}_{\text{contact}}$ with normal $\mathbf{n} = (\mathbf{P}_{\text{com}} - \mathbf{P}_{\text{contact}})/\text{d}$ and penetration distance $\delta$.
```

1.5. Force Model

Figure 4 shows the normal and tangential force models used in DEM simulations.

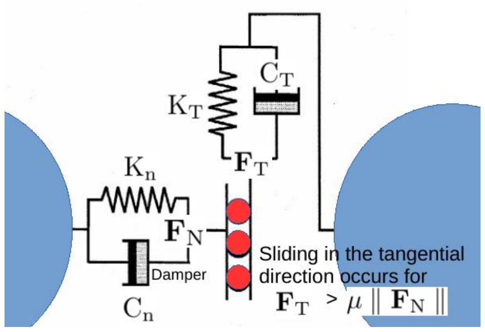

$\mathrm{C_T}$
$\mathrm{K_T}$
$\mathbf{F}_\mathrm{T}$
$\mathrm{K_n}$
$\mathbf{F}_\mathrm{N}$
Damper
$\mathrm{C_n}$
Sliding in the tangential direction occurs for
$\mathbf{F}_\mathrm{T} > \mu \parallel \mathbf{F}_\mathrm{N} \parallel$

Figure 4: Normal and tangential force models depicted by a spring dash-pot system.

A linear spring dash-pot model is used to calculate the normal force between particles given by :

$$\mathbf{F}_\mathrm{N} = (\mathrm{K_n}\delta)\bar{\mathbf{n}} - \mathrm{C_n}(\mathbf{V}_\mathrm{R}\cdot\bar{\mathbf{n}})\bar{\mathbf{n}}. \tag{1}$$

where δ is the penetration depth, $\mathbf{V}_\mathrm{R} = \mathbf{V}_1 - \mathbf{V}_2$ is the relative translational velocity, $\mathrm{K_n} = \frac{\mathrm{m_{eff}}}{\mathrm{t^2_{contact}}}\ln(\epsilon)^2 + \pi^2$ is the spring stiffness, $\mathrm{C_n} = \frac{2\ln(\epsilon)\sqrt{\mathrm{K_n m_{eff}}}}{\sqrt{\ln(\epsilon)^2+\pi^2}}$ is the viscous damping coefficient, $\bar{\mathbf{n}}$ the normal at contact, ϵ is the coefficient of restitution and $\mathrm{m_{eff}} = (\frac{1}{\mathrm{m_1}} + \frac{1}{\mathrm{m_2}})^{-1}$ is the effective mass of the particles. The contact time $\mathrm{t_{contact}}$ is determined by the properties of the material. However in most cases experimental data is not readily available for a particular material. For such cases $\mathrm{K_n}$ is chosen such that that physical quantities are conserved during integration for the typical range of velocities observed in tumbling mill simulations [4, 12, 21].

Typical DEM simulations use a spring stiffness and time-step that limits the maximum penetration depth to $\delta_{max} \leq 0.05\mathrm{r}$ where r is the radius of the smallest particle [13, 7, 22, 23].

Tangential Contact. In CPU DEM codes a linear spring dash-pot model is also used to calculate the tangential force given by:

$$\mathrm{F_T} = -\min\left[\mu \parallel \mathbf{F}_\mathrm{N} \parallel, (\mathrm{K_T}\int(\mathbf{V}_\mathrm{T}\mathrm{dt}) - \mathrm{C_T}\,\|\mathbf{V}_\mathrm{T}\|)\right], \tag{2}$$

where $\mathbf{V}_\mathrm{T} = (\mathbf{V}_\mathrm{R} - (\mathbf{V}_\mathrm{R}.\bar{\mathbf{n}})\bar{\mathbf{n}})$ is the relative tangential velocity, μ the coefficient of friction, $\mathrm{K_T}$ the tangential spring stiffness and $\mathrm{C_T} = \frac{2\ln(\epsilon)\sqrt{\mathrm{K_T}m}}{\sqrt{\ln(\epsilon)^2+\pi^2}}$ the tangential damping coefficient. The integral of the tangential velocity over the duration of the contact behaves as an incremental spring that stores energy from the relative tangential motions between the particles. This represents the elastic tangential deformation of the contacting particles while the dash-pot dissipates a proportion of energy. The magnitude of the tangential force is limited by $\mu \parallel \mathbf{F}_\mathrm{N} \parallel$ at which point the particles begin to slide over each other.

However, due to the sorting of data on the GPU [20], maintaining contact history for the duration of the collision is currently not possible to implement without severe performance penalties. Thus GPU codes use an history independent tangential model [13, 16, 22] that ignores the elastic tangential deformation in the tangential direction during contact and only accounts for energy dissipation. This makes GPU computations many times faster than CPU computations [13, 20, 24]. Therefore the GPU friction model is simplified to

$$\mathrm{F_T} = -\min\left[\mu \parallel \mathbf{F}_\mathrm{N} \parallel, \frac{\min\left[\min\left[\|\mathbf{V}_{1\mathrm{T}}\|, \|\mathbf{V}_{2\mathrm{T}}\|\right], \mu\,\|\mathbf{V}_\mathrm{T}\|\right]\mathrm{m_{eff}},}{\triangle\mathrm{t}}\right], \tag{3}$$

where $\mathbf{V}_{1\mathrm{T}}$ and $\mathbf{V}_{2\mathrm{T}}$ are the tangential velocities of each particle. This model has been shown to match experiment very well for simulations where the particles are constantly in motion in the previous works by the authors [20].

1.5.1. Calculation of power drawn by a mill

Grinding in the mineral processing industry is performed primarily by a ball mill which consumes a vast amount of energy and can account for as much as half of the processing cost. Thus understanding grinding mechanisms and estimating the power drawn by a mill can give guidance to improve the operational energy efficiency. The harsh environment inside the mill makes obtaining experimental data difficult. Thus the physical quantities calculated in a DEM simulation provide valuable insight to improve the efficiency of a mill. Since the power drawn by a mill is largely determined by the dynamics of the charge within the mill, we can obtain a good estimate of the power required by analyzing the energy loss mechanisms in a DEM simulation.

The total energy consumed by a mill is simply the net sum of the energy dissipated through contact $\mathrm{E}_{\mathrm{diss}} = \sum_i^{\mathrm{K}}(\| \mathbf{F}_{\mathrm{diss}} \| \triangle \mathrm{x})$ where K is the number of contacts . Here $\mathbf{F}_{\mathrm{diss}}$ is given by the damping and friction forces in Equations (1), (2) and (3) respectively and is assumed to be constant over the distance $\triangle$x, which the force acts. $\triangle$x is estimated by $\| \mathbf{V} \| \triangle \mathrm{t}$ since we do not store contact history in our GPU implementation. Thus the power consumed can be estimated by Power $= \frac{\mathrm{E}_{diss}}{\mathrm{t}}$ where t is the duration over which we wish to calculate the power for.

1.6. Calibration of model parameters.

In a DEM simulation the model parameters are chosen to either match experimental results or to reproduce a desired behavior. Tuning these parameters is a tedious task with the plethora of different models used in DEM simulations. Little guidance can be found in literature as the problems being simulated are quite often unique. In the area of ball mill simulations the 2D DEM code Millsoft developed by Rajamani et al. [1] was the first code to be employed in the simulation of mills and has been validated extensively over the past two decades. Thus we verify our GPU code using a non-incremental tangential force model, Equation (3) against Millsoft using an incremental tangential force model, Equation (2). In mill simulations the dynamics of the system is governed by the rotation speed Ω of the lifters. The distance covered by a lifter during a time-step is given by $\mathrm{x}_{\mathrm{Lifter}} = \mathrm{R}.\omega.\triangle \mathrm{t}$ where R is the radius of the mill drum and $\omega = \frac{\pi}{30}.\Omega$ is the angular velocity of the mill. Thus the maximum time-step for a mill rotating Ω rpm than yields a maximum penetration distance $\leq \delta_{max}$ is given by:

$$\triangle \mathrm{t} = \frac{\delta_{max}}{\mathrm{R}.\omega} \tag{4}$$

1.6.1. Effect of parameters on charge profile

In this simulation we use a mill with diameter of 516 cm and length set equal to the ball diameter for simulating motion in 2D when using a 3D code. Thirty two rows of rectangular lifters with a height of 9.5 cm and width of 15 cm is used with the mill rotating at 14 rpm. We attempt to tune the model parameters of the GPU code to reproduce the charge profile obtained using Millsoft. We start of by using the same parameters with the GPU code as given in Table 1. Note that the maximum allowed time-step given by Equation (4) is$\triangle \mathrm{t}_{\mathrm{max}} = 1.65 \times 10^{-5}$ using a maximum penetration distance of 0.025r .

Table 1: Model Parameters used in simulation for a 2D mill.

Parameter	$\mathrm{K}_{\mathrm{n}}(\mathrm{N.m}^{-1})$	ϵ	K_{T}	μ	$\triangle$t
GPU	4×10^5	0.45	-	0.70	1×10^{-5}
CPU	4×10^5	0.45	3×10^5	0.70	1×10^{-5}

Figure 5(a) shows the charge profile obtained with Millsoft and sub-figures (b)-(d) the GPU profiles for various values of μ. We notice in Figure 5(b) using the same frictional value that there is a good match with the shoulder position and the release point of the lifters. However there is a significant difference with the toe position in the GPU simulation at 6 30' while it is at 7 30' in the CPU simulation. This difference is attributed to the absence of a restorative force in the GPU tangential model, which results in a greater resistive force being experienced by the particles for the same value of μ. This keeps the center of mass of the distribution at a higher point as illustrated by the belly position. In Figure 5(c) we use a lower friction value of $\mu = 0.6$ in the GPU simulation. We notice that the toe is lower but the frictional force is still too high. In Figure 5(d) we reduce the friction value to $\mu = 0.4$ in the GPU simulation. We now see a much better agreement of the toe, shoulder and belly positions.

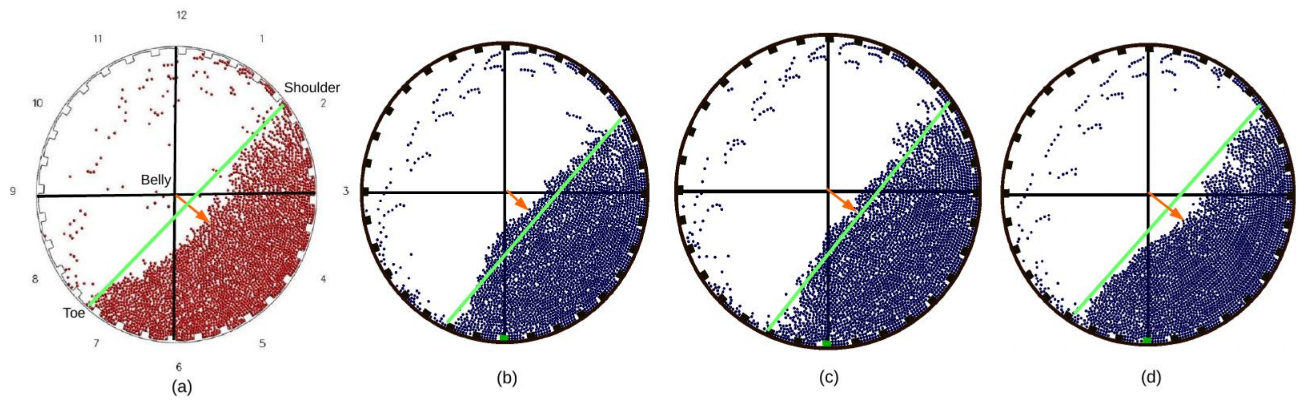


Figure 5: Charge profiles for CPU (a) $\mu = 0.70$ and GPU (b) $\mu = 0.70$ (c) $\mu = 0.60$ and (d) $\mu = 0.40$, N= 2916 (radius=2.5 cm).

The second parameter that we can tune is the coefficient of restitution ϵ . In Figure 6 we vary the value of ϵ while maintain the frictional value of $\mu = 0.4$. We notice that for $\epsilon = 0.25$ (Figure 6(a)) the distribution is packed much tighter as larger fraction of energy is lost during contact. For $\epsilon = 0.65$ (Figure 6(c) the packing is less dense producing a greater dispersion when particles are released that provides a closer match to the distribution obtained with Millsoft (Figure 5(a)).

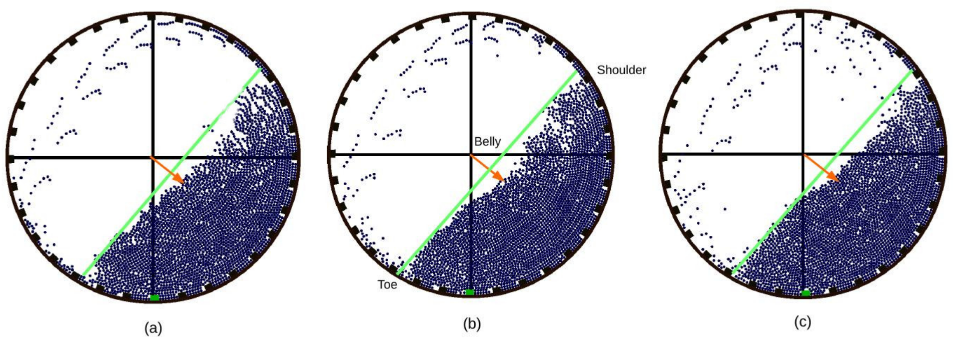


Figure 6: GPU charge profiles for (a) $\epsilon = 0.25$ (b) $\epsilon = 0.45$ and (c) $\epsilon = 0.65$, N= 2916 (radius=2.5 cm).

Now that we have calibrated the GPU parameters, we vary the number of particles (N) and compare the charge profiles using the same parameters as summarized in Table 2.

Table 2: Model Parameters used in simulation for a 2D mill.

Parameter	$K_n(N.m^{-1})$	ϵ	K_T	μ	$\triangle t$
GPU	4×10^5	0.65	-	0.40	1×10^{-5}
CPU	4×10^5	0.45	3×10^5	0.70	1×10^{-5}

The charge profiles for N= 5344 are compared in Figure 7, while Figure 8 compares the charge profiles for N= 11664. We notice a good match using the same set of parameters.

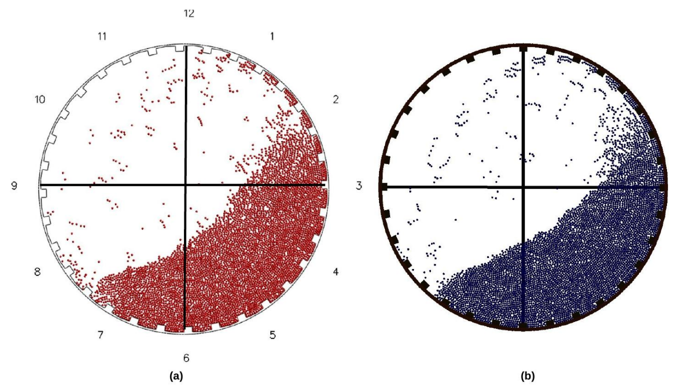


Figure 7: (a) CPU and (b) GPU charge profiles. N= 5344 (radius=1.85 cm).

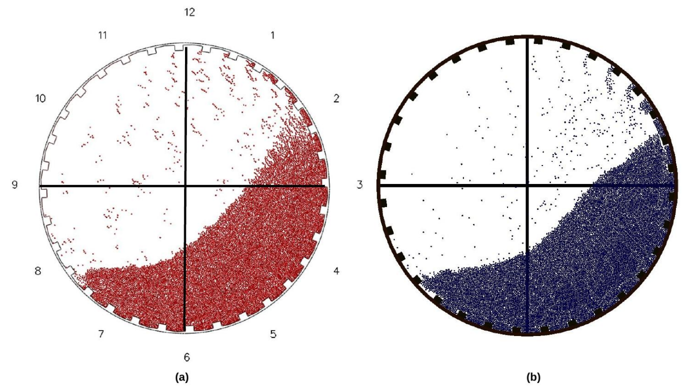


Figure 8: (a) CPU and (b) GPU charge profiles. N= 11664 (radius=1.25 cm).

2. Experimental validation of GPU DEM for mill simulations.

2.1. Three-dimensional mill

We use the experimental data obtained by Venagopal and Rajamani [4] using a 90 cm diameter by 15.0 cm length mill containing eight 4.0 cm square lifters. The face of the mill was made of Plexiglas$^{\mathrm{TM}}$ as to enable photographing of the tumbling charge, with the shell and lifters being made of steel (density of 7800kg.m^{-3}). The mill was operated at 30%, 50% and 70% of critical speed for two levels of mill filling, 20% and 30% by volume respectively using steel balls with a radius of 2.5cm. The critical speed of a mill is the speed at which particle motion changes from cascading to centrifuging. Note that the maximum allowed time-step given by Equation (4) is $\triangle t_{\max} = 4.14 \times 10^{-5}$ using a maximum penetration distance of 0.025r.

2.1.1. Calibration of model parameters

In the previous simulation we studied the effect of parameters on the charge profile in the mill. In this simulation we investigate the effect of the parameters on power draw as well for a mill with 20% loading at a mill speed of 32 rpm (70% of critical speed) drawing 532W of power. We firstly match the charge profile by varying the frictional value as we saw in the previous simulation that this had the largest effect on the charge profile. Figure 10 shows the charge profile for various values of μ. We notice that the shoulder, toe and belly positions are lower as expected as we decrease the value of μ. Figures 9 (f) and (g) having values of $\mu = 0.20$ and $\mu = 0.15$ respectively gives the best match to the experimental profile depicted in Figure 9 (a). Thus the ideal value of μ seems to be in the range [0.15 : 0.20]

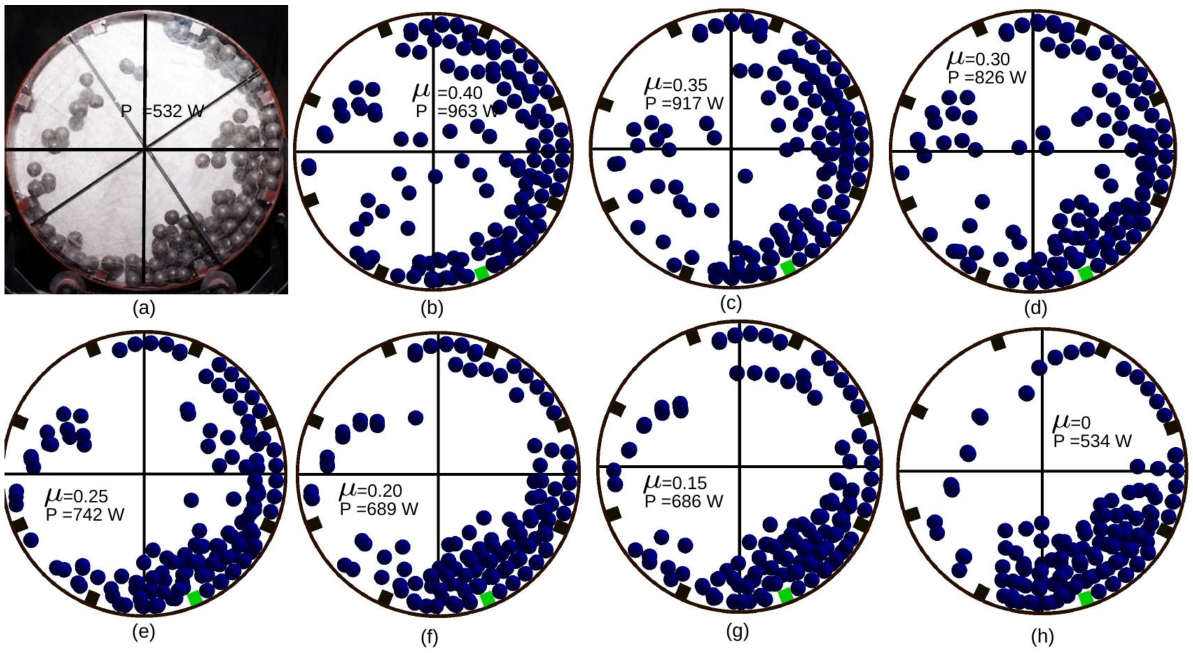


Figure 9: (a) Experiment (b) GPU charge profiles. N= 168 (20% filling), rpm = 32 (70% critical speed), varied μ,$\epsilon = 0.65$.

Figure 10 depicts the charge profile for varying values of ϵ using $\mu = 0.20$. We firstly notice that there is little change with the shoulder and toe positions with the belly becoming less dense as we increase ϵ . The ideal value of ϵ seems to be in the range [0.80 : 0.85] as power at the limits of the range bounds the experimental power of 532W.

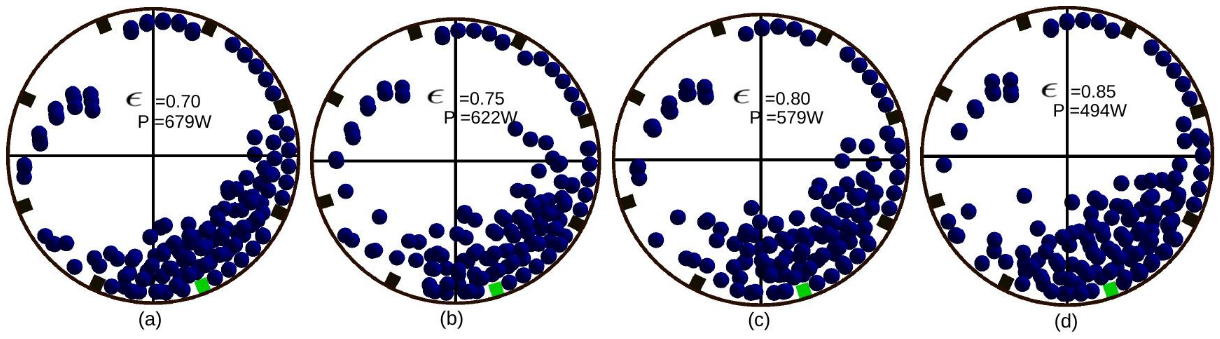


Figure 10: (a) Experiment (b) GPU charge profiles. N= 168 (20% filling), rpm = 32 (70% critical speed), varied ϵ.

Now that we have a reasonable match for the charge profile we tune the parameters to yield the desired power values. We use a finite number of combinations of μ and ϵ in the ranges that we deemed to yield the best charge profiles in Figure 9 and 10. Table 3 contains the four combinations we use as well as the average and maximum errors obtained for mill speeds of 14, 22 and 32 rpm. Ideally the range of values should be more than the four we have chosen to get the best possible match. However we only wish to show the effect of DEM parameters on mill simulations rather than do a detailed calibration of DEM parameters. Combination 3 gives the lowest error and will thus be used to predict power and charge profiles for the rest of this section.

Table 3: Average error for various parameter combinations for a 3D mill.

Combination	ϵ	μ_T	Max Error	Average Error
1	0.80	0.15	7.88%	7.09%
2	0.825	0.20	7.34%	5.23%
3	0.825	0.15	5.99%	4.61%
4	0.85	0.175	8.01%	5.77%

2.1.2. Charge motion and power draw

Representative snapshots of the GPU DEM predicted charge profiles alongside the still camera images for each of the experiments are shown in Figures 11 to 14 with the associated power draft in Table 4. Charge profiles predicted by the GPU DEM code are consistent with observed charge profiles. The positions of the toe and shoulder of the charge are also reasonable with a slightly lower toe position and lower mass distribution which is expected due to the simpler tangential model used. An exact match between charge trajectories is difficult to obtain due to simplifying assumptions made in the DEM model as well as the mechanical losses and the geometry of the mill not being exactly the same due to wear and manufacturing processes.

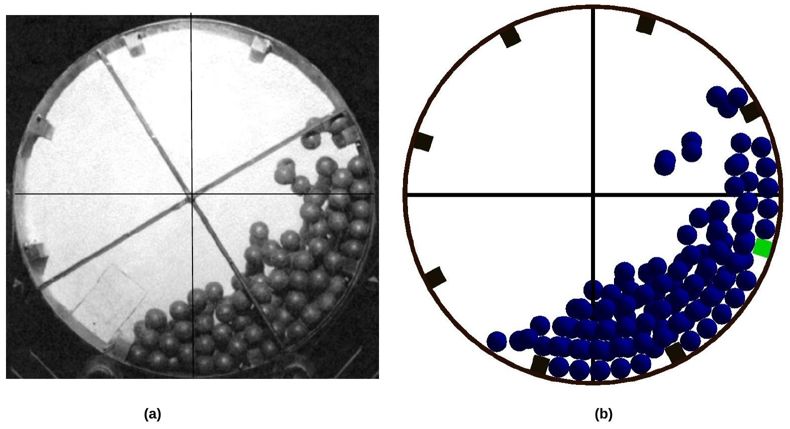


Figure 11: (a) Experiment [4] (b) GPU charge profiles. N= 168 (20% filling), rpm = 14 (30% critical speed).

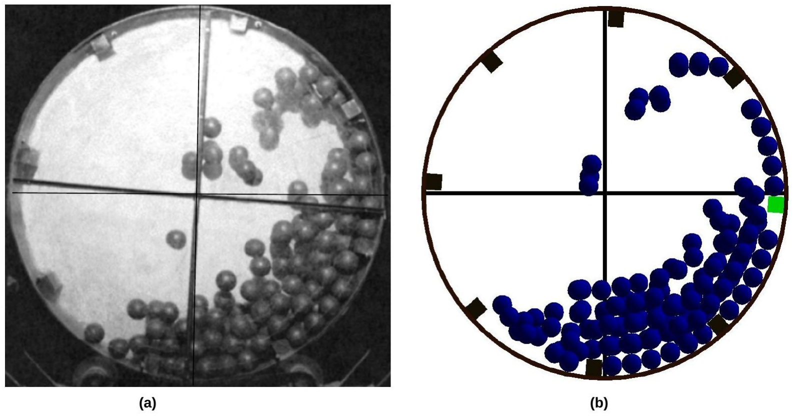


Figure 12: (a) Experiment [4] (b) GPU charge profiles. N= 168 (20% filling), rpm = 22 (50% critical speed).

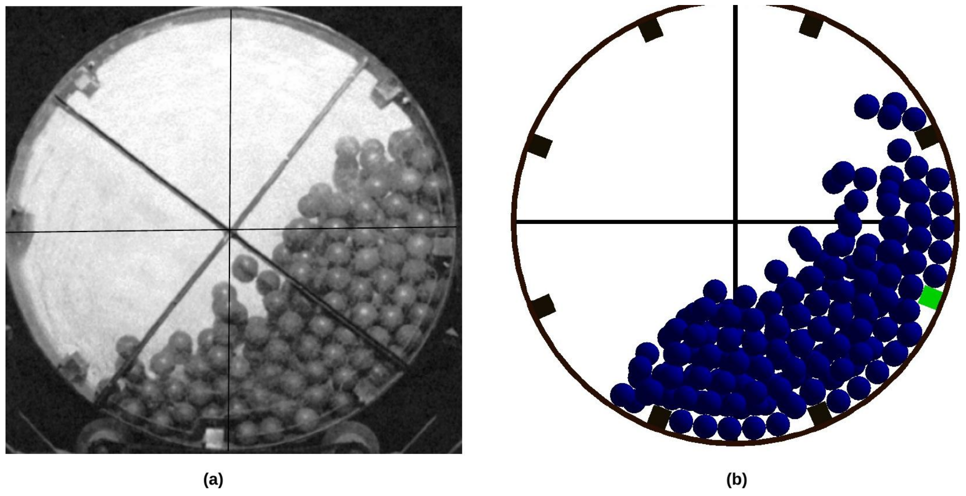


Figure 13: (a) Experiment [4] (b) GPU charge profiles. N= 243 (30% filling), rpm = 14 (30% critical speed).

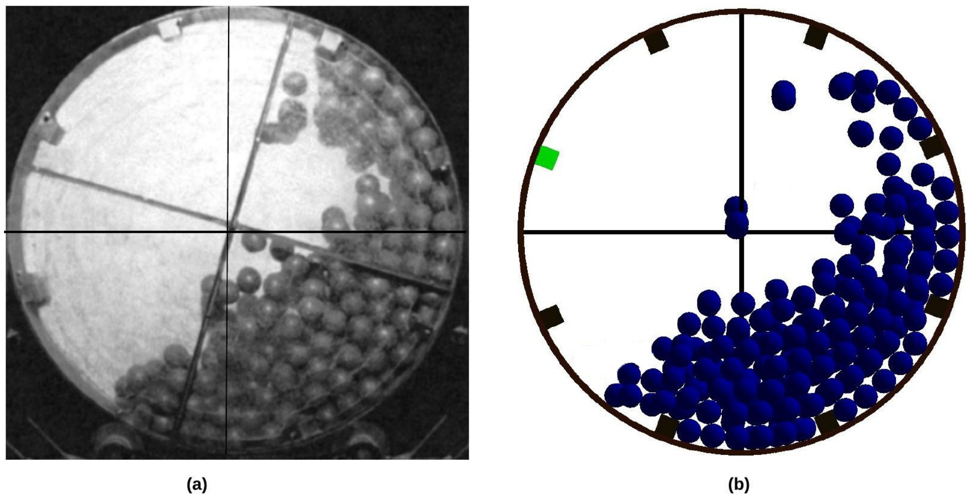


Figure 14: (a) Experiment [4] (b) GPU charge profiles. N= 243 (30% filling), rpm = 22 (50% critical speed).

Table 4 depicts the power values obtained. We see a good match for the 30% loading scenario as the error is only 4.07%. Note that we only tuned the parameters for the 20% loading scenario. This gives confidence in the selection of the model parameters as we can predict the effects of mill-load on the power draw.

Table 4: Power draw with experiment and GPU DEM for 3D mill.

RPM	Filling (20%)		Filling (30%)	
	Power(w)			
	Experiment	GPU DEM	Experiment	GPU DEM
14	301	336	393	409
22	459	437	617	636
32	532	520		

2.2. Charge motion and power draw for a slice mill

In this simulation we use the experimental data as given by Moyes et al. [25] for a pilot mill with a diameter of 55cm and length of 2.35cm containing twelve rows of 22 cm square lifters. The mill is loaded to

25% and 35% respectively with steel balls having a radius of 2.2 cm and density of 7800kg.m^{-3}. For their DEM simulations Moyes et al. use the same model parameters for different mill-speeds. However as discussed in Section 1.6 the model parameters are determined by the rotation speed of the mill. The maximum allowed time-step using Equation (4) for a maximum penetration distance of 0.10r per step at 160% of critical speed (93.30) RPM is $\Delta t_{max} = 4.14 \times 10^{-6}$. To allow comparison with published results, we use the same Δt for increasing speed but include results using a time-step given by Equation (4) as mill speed increases. Table 5 summarizes the model parameters used.

Table 5: Model Parameters used in simulation for slice mill.

Parameter	$K_n(N.m^{-1})$	$\epsilon_{particle}$	ϵ_{shell}	K_T	$\mu_{particle}$	μ_{shell}	Δt
GPU	4×10^5	0.85	0.80	-	0.15	0.20	2×10^{-5}
CPU	4×10^5	0.66	0.36	4×10^5	0.14	0.39	2×10^{-5}

Figure 15 shows how the power varies as a function of rotation speed. We see a good match for sub-critical speeds which is the normal operation mode of a mill between both DEM codes and experiment. At super-critical speeds there is a slight difference due to using the CPU or GPU codes with a constant time step. However the results using the time-step given by Equation (4) as we increase mill speed shows a better match to experiment.

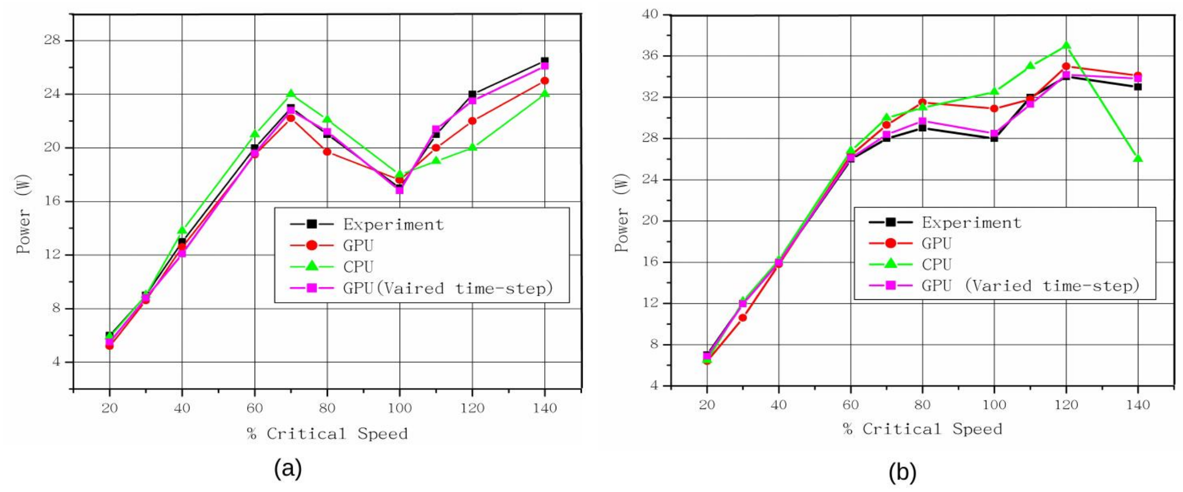


Figure 15: Power draw for (a) 25% and (b) 35% loading between experiment [25], GPU and CPU simulations.

Figures 16 and 17 depicts the charge profile for sub-critical speeds. We see a good match which is expected as the power values are similar. Note that it is difficult to do an accurate frame matching.

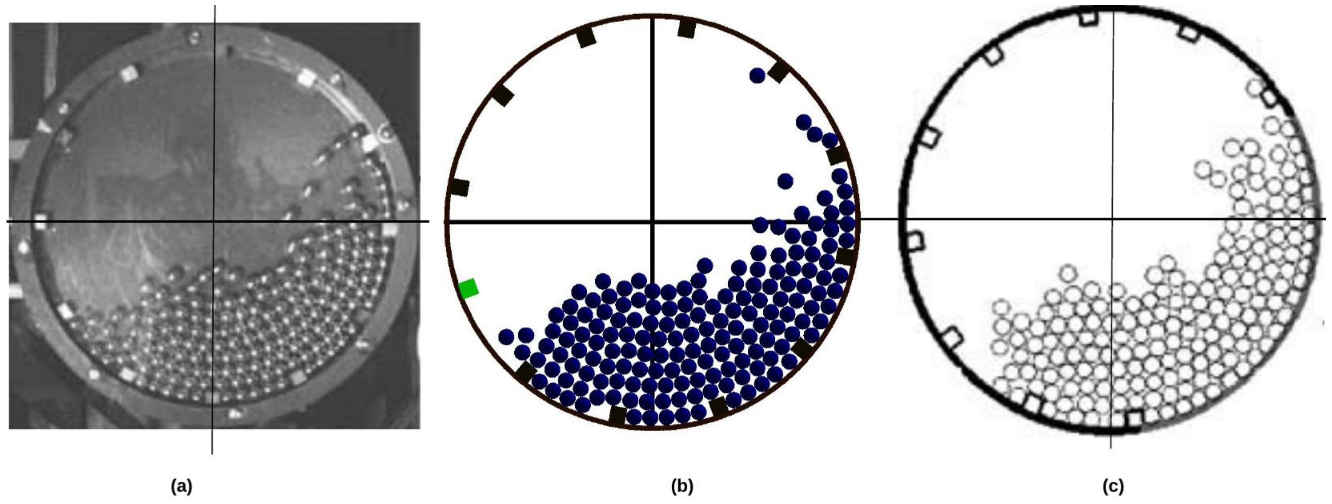


Figure 16: (a) Experiment [25] (b) GPU and (c) CPU charge profiles. N= 169 (35% filling), rpm = 17.50 (30% critical speed).

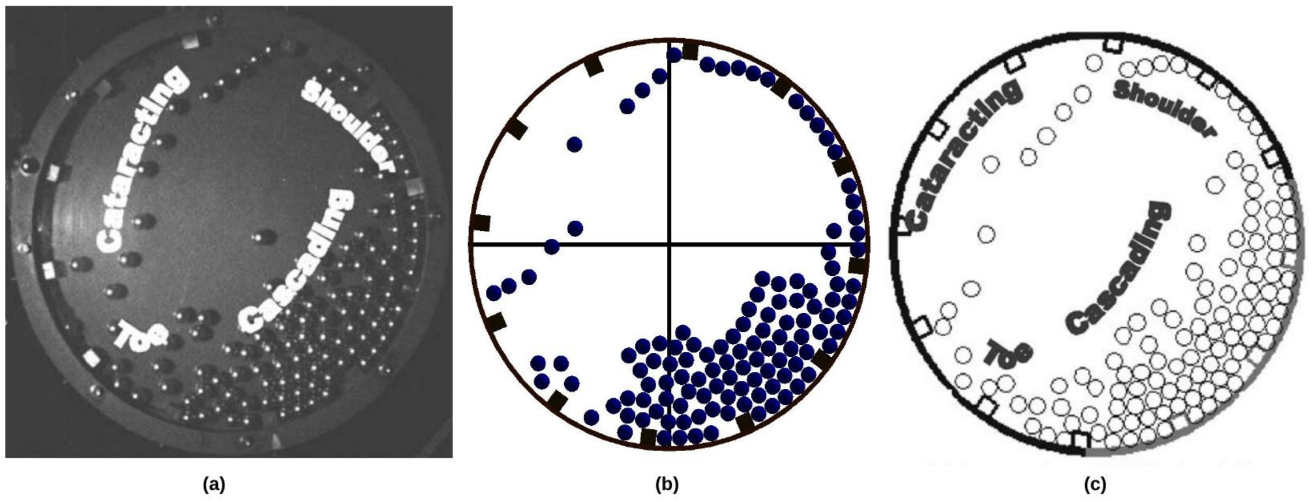


Figure 17: (a) Experiment [25](b) GPU and (c) CPU charge profiles. N= 120 (25% filling), rpm = 40.81 (70% critical speed).

Figures 18 and 19 depict the charge profile for super-critical speeds. We note that DEM correctly predicts 1 and 2 layers of centrifuging particles for 100 and 160 percent of critical speed respectively.

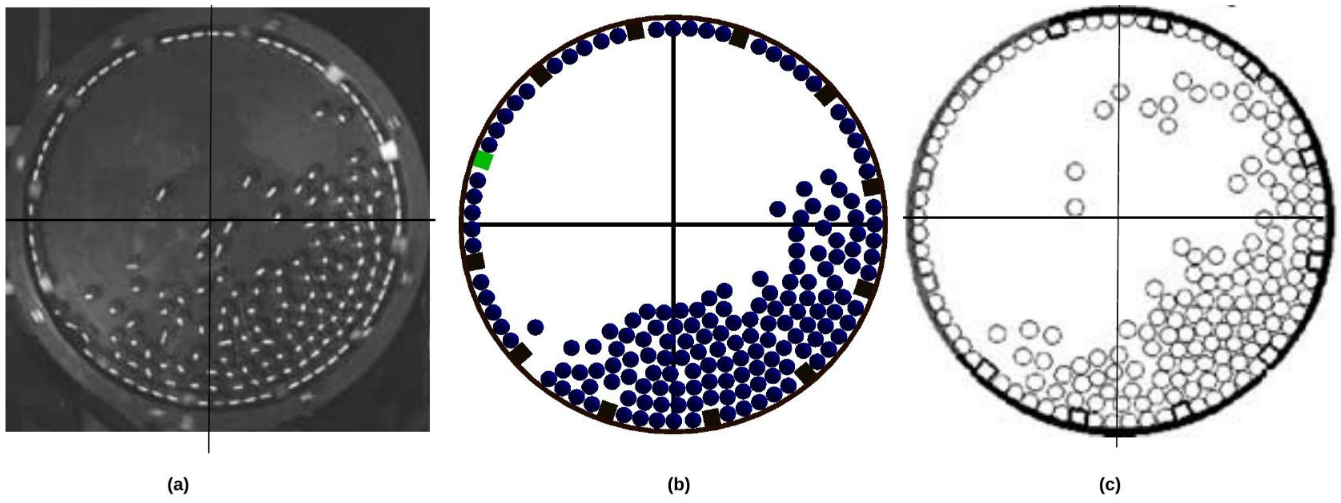


Figure 18: (a) Experiment [25] (b) GPU and (c) CPU charge profiles. N= 169 (35% filling), rpm = 58.30 (100% critical speed).

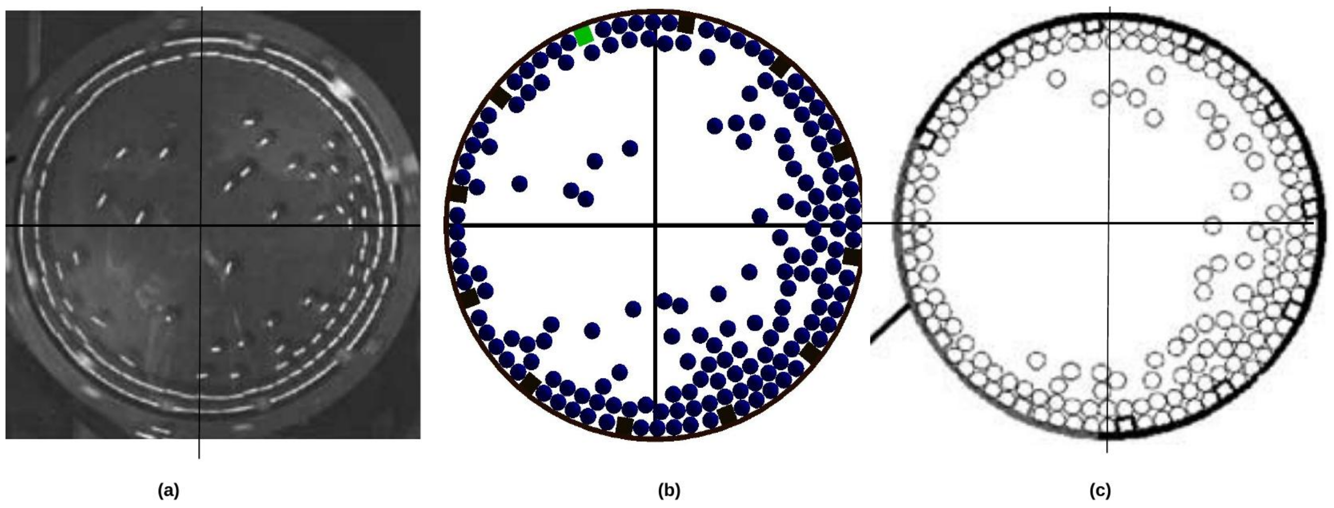


Figure 19: (a) Experiment [25] (b) GPU and (c) CPU charge profiles. N= 169 (35% filling), rpm = 93.30 (160% critical speed).

3. Industrial Mill Simulation

The Los Bronces SAG mill is a 10.12m diameter by 4.7m long mill rotating at 10 rpm . The mill charge is made up of 31% ore and 10% balls by volume. The power draft reported for this mill (Malcolm et al.) is 7.1 MW. Since, no further information about mill internals was available, typical values of operating parameters for this type of mill was used. It is presumed that the mill would be fitted with 64 rows of high low lifters as shown in Figure 20.

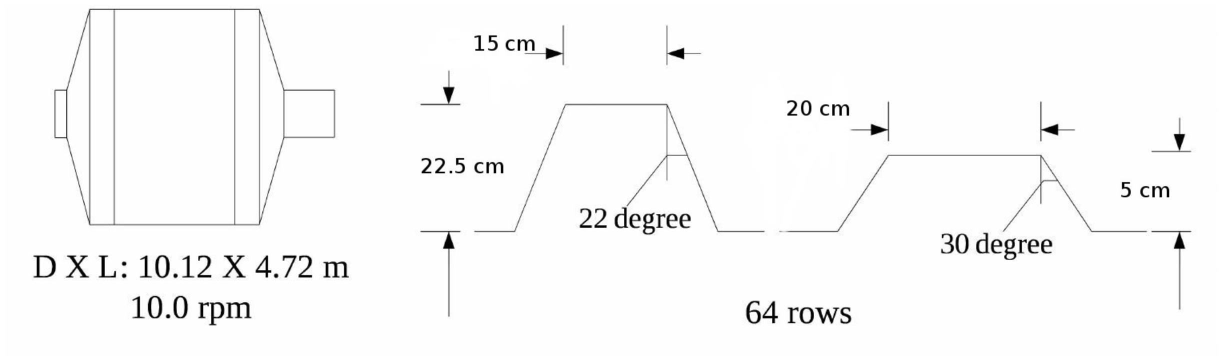


Figure 20: Lifter design Los Bronces semi autogenous mill.

Equilibrium ball size distribution with top ball size of 12.5 cm was used in the simulation. The ore in the mill charge was approximated as a Gaudin-Schuhmann [26] distribution with slope 0.6 and top size of 15.0 cm. The combined weight distribution and number of ore and ball particles in the charge mix as shown in Table 6. The aim of this study is to do a qualitative investigation as opposed to a quantitative investigation due to the uncertainties in the experimental setup. We investigate how the power consumption varies over revolutions, as well as how the power gets dissipated inside the mill.

Table 6: Charge distribution for Los Bronces mill.

Diameter(cm)	Density(kg/m^3)	Weight (%)	Number
	ORE		
15.0	2850	9	6320
12.5	2850	9	9020
10.1	2850	30	81881
	BALLS		
12.4	7800	21	9800
10.0	7800	19	16171
8.7	7800	12	16220
TOTAL			**139392**

Table 7 summarizes the model parameters used in the simulations in Section 3.

Table 7: Model Parameters used in simulation of Los Bronces mill.

Parameter	$K_n(N.m^{-1})$	$\epsilon_{particle}$	ϵ_{lifter}	K_T	$\mu_{particle}$	μ_{lifter}	$\triangle t$
GPU	1.5×10^6	0.75	0.75	-	0.40	0.70	2×10^{-5}

Figure 21 shows the charge profile in the mill. The charge profile is what one would anticipate, for such large filling of 41%. The charge shoulder is nearly at 2 o'clock and the toe is between 7 and 8 o'clock points on the mill circle. We also see radial segregation with the smallest particles moving to the mill shell as predicted by theory and experiment.

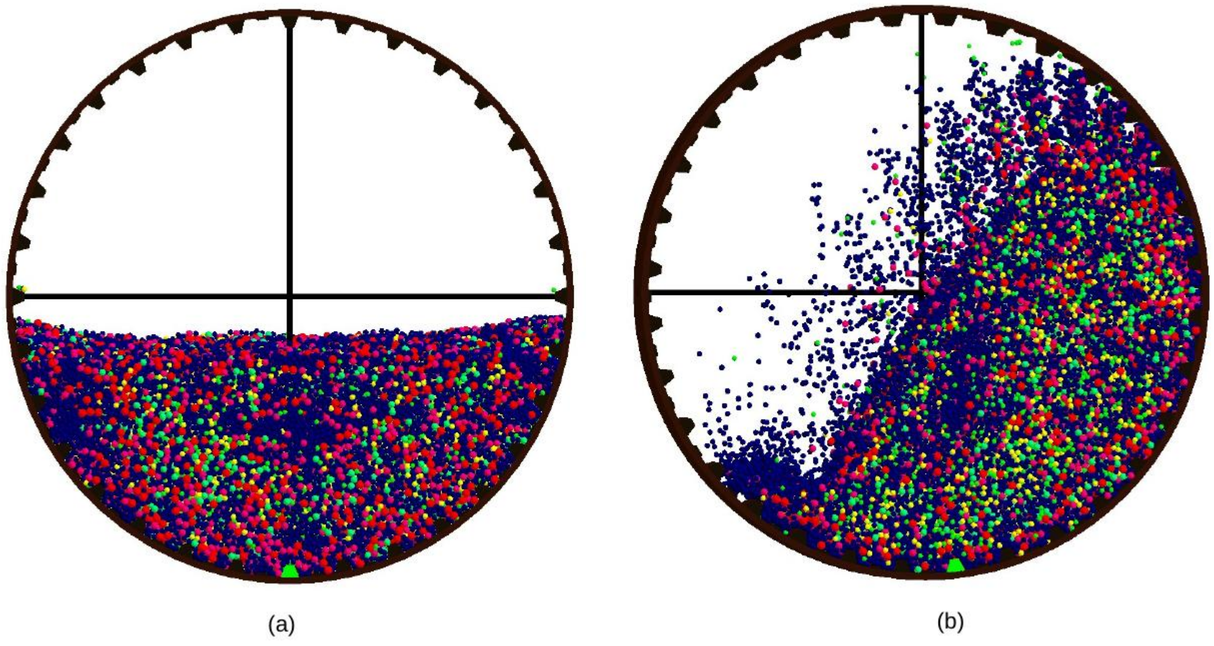


Figure 21: (a) Initial conditions N= 139392 (41% filling) (b) Charge profile of Los Bronces mill.

The computed power seems to stabilizes after six revolutions as shown in Figure 22 (a). The large number of spheres in the simulation smooth out the energy consumed in collisions per revolution. Hence, accurate dependence of contact parameters on contact velocity is unnecessary here. The computed power consumption is 6.8MW, which is slightly lower that the experimental of 7.1MW. Note, this value is arbitrary as we showed we could match the power consumption exactly by changing the coefficient of restitution. The important observation is how the power consumption varies over revolutions, as well as how this power is dissipated. Figure 22 (b) shows the contribution to power draw by particle to particle collisions, particle to mill shell collisions and particle to lifter collisions. Around 68% of the power draw comes from the lifter since it lifts the majority of the load to the shoulder of the mill and hence excessive forces occur at the points of contact on the lifters. Likewise, around 18% is contributed by the mill shell as it supports the load between two lifters. Thus less than 14% is contributed by particle-particle collision. In addition, as the particle lifter power decreases slightly over revolutions, the mill-shell power draw increases slightly. The particle-particle power dissipation remains virtually constant over the revolutions.

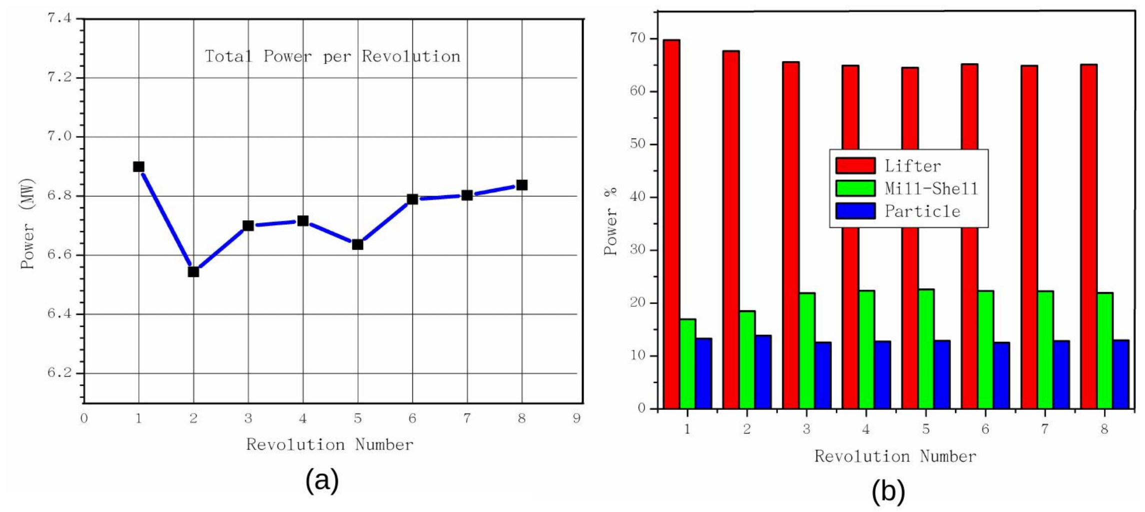


Figure 22: (a) Total power draw (b) Power distribution over time of Los Bronces mill.

3.1. Performance scaling

To gauge the performance of our code we increased the length of the mill to 2800 cm to accommodate four million mono-sized steel balls with a diameter of 6 cm. Figure 23 shows the charge profile.

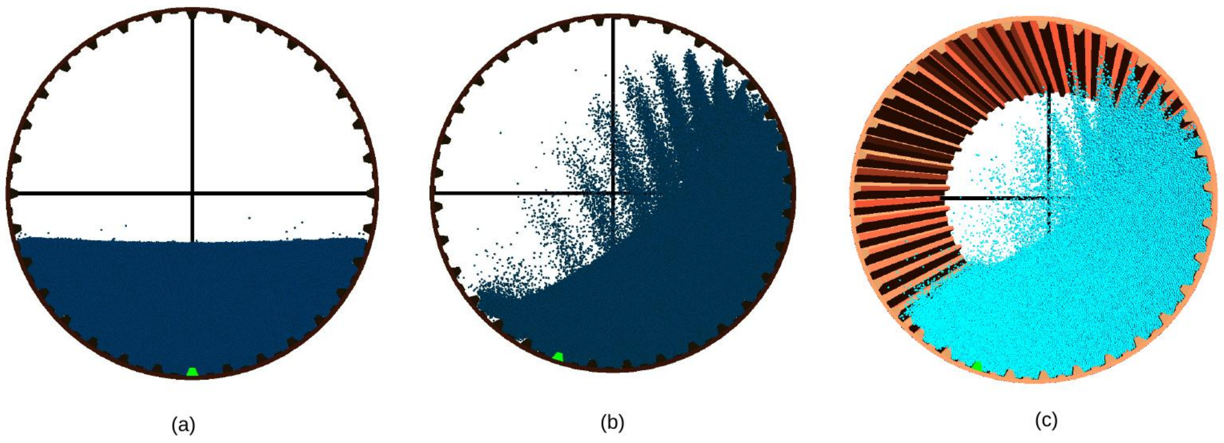


Figure 23: (a) Initial conditions N=4 $\times$ 10^6 (35% filling) (b) steady state profile (orthogonal view) (c) steady state profile (isometric view) .

The power draft is distributed with 47% to particle-particle collisions, 44% to particle-lifter collisions and 9% to particle mill-shell collisions. The shear number of particles simulated results in particle-particle collisions consuming the most amount of energy. The GPU compute time for one revolution (6 seconds) using a time-step of 2×10^{-5} with a NVIDIA Kepler GPU is 7 hours (12 FPS). Figure 24 shows the scaling of our code with increased particle number, we see a trend of linear scaling which is good indicating of the scalability of our code. The GPU compute time for one revolution (6 seconds) of a simulation consisting of 10 million particles using a time-step of 2×10^{-5} with a NVIDIA Kepler GPU will take just 18.5 hours with a simulation of 100 million particles taking just over a week.

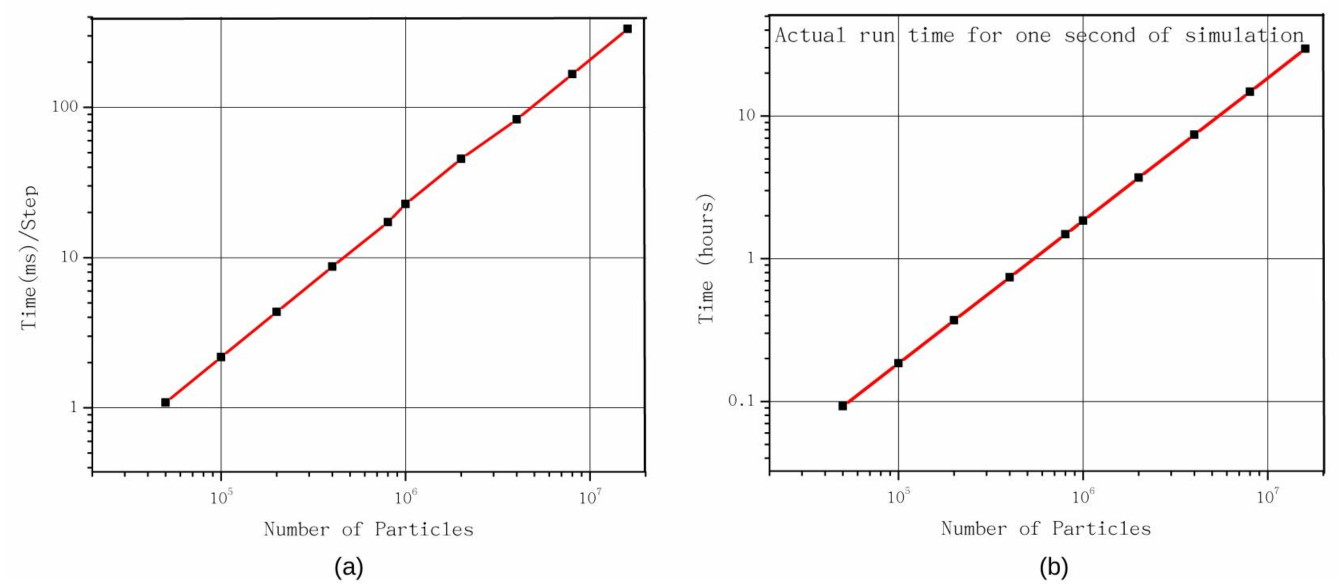


Figure 24: Scaling of GPU code with number of particles (N) .

To show the benefits of a full 3D simulation we performed the same simulation in 2D with the same parameters and obtained a very different charge profile as depicted in Figure 25 (a). Figure 25 (b) depicts the charge profile for a slice of 10% length. We notice that the pattern is very similar to the full length of the mill. Clearly the effect of the axial direction cannot just be neglected as the 2D case cannot fully reproduce the dynamics.

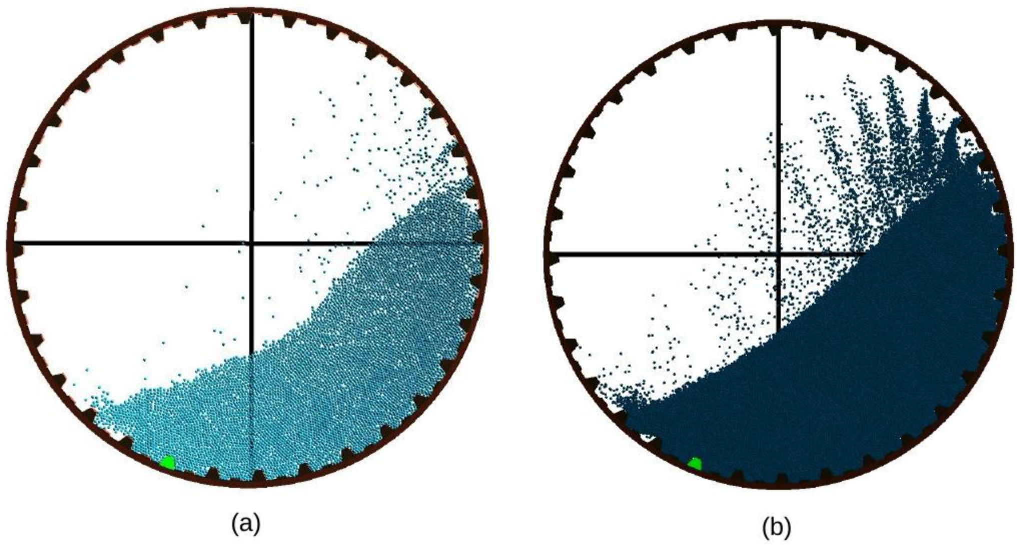


Figure 25: (a) 2D steady state profile, N=6744 (b) Steady state profile for a slice 10% of the length, N=385534.

4. Conclusions

In this paper we have presented a novel approach for modeling tumbling mills utilizing the GPU architecture. The modular approach of our code allows us to analyze the distribution of power amongst different collision types in the mill amongst other things which lends insight that may be exploited to improving the energy efficiency of mills. We achieve a new performance level in DEM modeling of mills by simulating 16 million particles at 3 FPS on a laptop GTX 880 GPU. Our code can handle 1 billion particles using the 12GB of memory available on a NVIDIA K40 GPU. However such a large simulation should ideally be run with multiple GPUs and is currently under development.

5. Acknowledgments

The first author acknowledges the support of the university of Utah in providing research funds for his stay at the university to do this work. We gratefully acknowledge the support of NVIDIA Corporation with the donation of the GPU used for this research.

References

[1] B. Mishra, R. Rajamani, Numerical simulation of charge motion in a ball mill., 7th European Symposium on Comminution. 1 (1990) 555–563.

[2] M. Powell, The effect of liner design on the motion of the outer grinding elements in a rotary mil, International Journal of Mineral Processing 31 (1991) 163–193.

[3] M. Dennis, R. Rajamani, Evolution of the perfect simulator, International Autogenous and Semiautogenous Grinding Technology Proceedings 31 (2001) 163–193.

[4] R. R. Venugopal, R., 3d simulation of charge motion in tumbling mills by the discrete element method., Powder Technology 115 (2001) 157–166.

[5] B. K. Mishra, R. Rajamani, Three dimensional simulation of plant size sag mills, International Conference on Autogenous and Semiautogenous Grinding Technology 31 (2001) 48–57.

[6] J. Herbst, L. Nordell, Optimization of the design of sag mill internals using high fidelity simulation, International Conference on Autogenous and Semiautogenous Grinding Technology 31 (2001) 150–164.

[7] P. W. Cleary, Recent advances in dem modelling of tumbling mills, Minerals Engineering 14 (2001) 1295–1319.

[8] R. Morrison, P. W. Cleary, Towards a virtual comminution machine. minerals engineering, Minerals Engineering 21 (2008) 770–781.

[9] P. W. Cleary, R. Morrison, Particle methods for modelling in mineral processing, International Journal of Computational Fluid Dynamics 23 (2009) 137–146.

[10] J. Alatalo, K. Tano, Comparing experimental measurements of mill lifter deflections with 2d and 3d dem predictions, DEM5 Proceedings, Queen Mary University, London, UK, 1 (2010) 194–198.

[11] J. Favier, Edem (2014).
URL `http://www.dem-solutions.com/`

[12] 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.

[13] J. Longmore, P. Marais, M. Kuttel, Towards realistic and interactive sand simulation: A gpu-based framework, Powder Technology 235 (2013) 983–1000.

[14] T. Harada, GPU Gems 3: Real-time rigid body simulation on GPUs, Vol. 3, 2008.

[15] R. Rajamani, S. Callahan, J. Schreiner, DEM Simulation of Mill Charge in 3D via GPU computing, Proceeding of the SAG conference, Vancouver, 2011.

[16] N. Govender, D. Wilke, S. Kok, Collision detection of convex polyhedra on the nvidia gpu architecture for the discrete element method, Journal of Applied Mathematics and Computation http://dx.doi.org/10.1016/j.amc.2014.10.013.

[17] NVIDIA, Nvida kepler gk110 architecture whitepaper (2012).
URL `http://www.nvidia.com/NVIDA KEPLER GK110 Architecture Whitepaper`

[18] J. Sanders, E. Kandrot, CUDA by example, Vol. 12, 2010.

[19] P. Cundall, Strack, A discrete numerical model for granular assemblies, Geotechnique 29 (1979) 47–65.

[20] N. Govender, D. Wilke, S. Kok, R. Els, Development of a convex polyhedral discrete element simulation framework for nvidia kepler based gpus, Journal of Computational and Applied Mathematics 270 (2014) 63–77.

[21] B. Mishra, R. Rajamani, Simulation of charge motion in ball mills. part 1: experimental verifications, Int. J. Mineral Process 40 (1994) 171–186.

[22] N. Bell, Y. Yu, Particle-based simulation of granular materials, Eurographics/ACM SIGGRAPH Symposium on Computer Animation 25 (2005) 29–31.

[23] D. Zhao, E. Nezami, Y. Hashash, J. Ghaboussi.J., Three-dimensional discrete element simulation for granular materials, Computer-Aided Engineering Computations: International Journal for Engineering and Software 23 (2006) 749–770.

[24] G. Neubauer, C. A. Radeke., Gpu based particle simulation framework with fluid coupling ability (March 2014).
URL `http://on-demand.gputechconf.com/gtc/2014/poster/pdf/P4143`

[25] O. Hlungwani, J. Rikhotso, H. Dong, M. Moys, Further validation of dem modeling of milling: effects of liner profile and mill speed, Minerals Engineering 16 (2003) 993–998.

[26] A. Macias-Garcia, Eduardo, M. Cuerda-Correa, M. Diaz-Diez, Application of the rosin-rammler and gates-gaudin-schuhmann models to the particle size distribution analysis of agglomerated cork, Materials Characterization 52 (2004) 159–164.