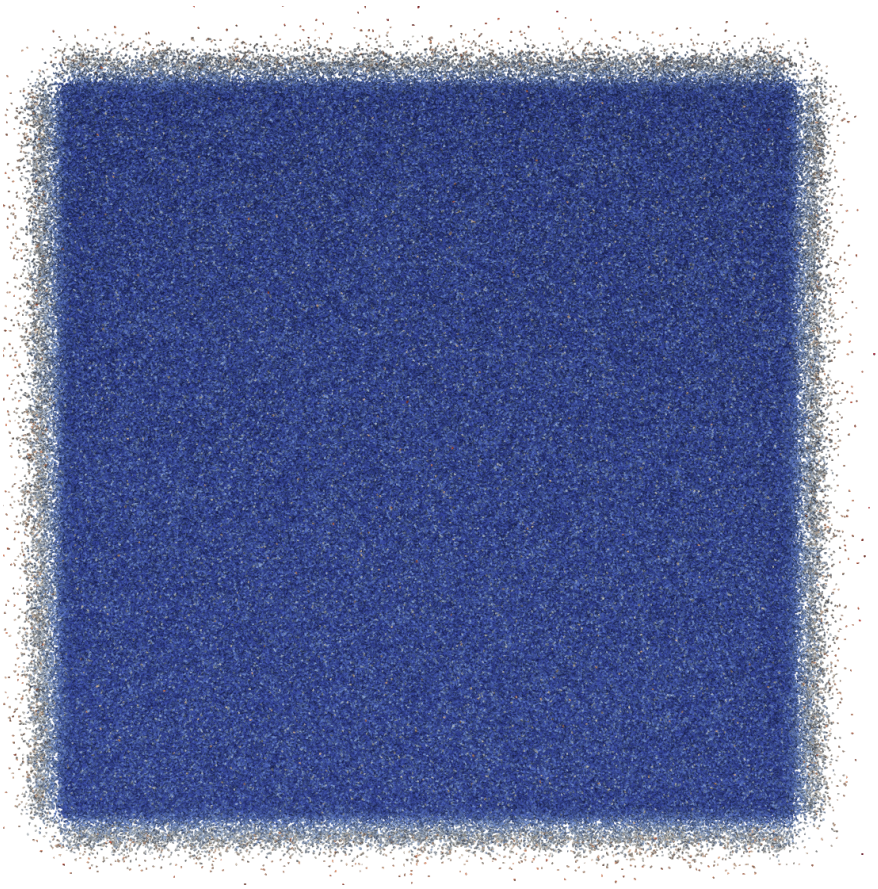
CHALMERS
UNIVERSITY OF TECHNOLOGY



A GPU Polyhedral Discrete Element Method

Formulation and implementation of large scale simulations for non-spherical particles using novel GPU techniques

Master's thesis in Complex Adaptive Systems

ADAM BILOCK

Department of Mathematical Sciences
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2020

MASTER'S THESIS 2020

A GPU Polyhedral Discrete Element Method

Formulation and implementation of large scale simulations for irregular non-convex particles using novel GPU techniques

ADAM BILOCK

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Mathematical Sciences
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2020

A GPU Polyhedral Discrete Element Method
Formulation and implementation of large scale simulations for irregular non-convex particles using novel GPU techniques
ADAM BILOCK

Supervisor: Klas Jareteg, Fraunhofer-Chalmers Research Centre
Examiner: Anders Logg, Department of Mathematical Sciences

Master's Thesis 2020
Department of Mathematical Sciences
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: One million Schönhardt polyhedral particles simulated with the presented method.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2020

A GPU Polyhedral Discrete Element Method
Formulation and implementation of large scale simulations for irregular non-convex particles using novel GPU techniques
ADAM BILOCK
Department of Mathematical Sciences
Chalmers University of Technology

Abstract

This thesis presents a Discrete Element Method (DEM) to simulate irregular shaped particles by a non-convex polyhedron representation. By using novel GPU techniques and an efficient HPC implementation the presented method shows a level of throughput not previously attained with polyhedron particle representations in the open literature. Further, via such a representation the exact volumetric overlaps of the particles are resolved and, as a result, the method is robust and numerically stable with respect to geometric changes. The efficient and well-behaved method allows for significant progress in the study of granular materials, where previously mainly the inadequate particle representation of spherical or clumped spherical particles have been used.

The exact volumetric overlaps are resolved by a simplex representation which allows for the use of non-convex particles without any decomposition, aiding both performance and the ease of use of the method. Further, care is given to attain efficient scaling of the method with respect to particle resolution. Such a property enables studies on higher resolution particles than previously shown in related work, and is result of efficient filtering of polyhedron triangles in the narrow contact phase. In addition, other novel techniques, such as a GPU BVH implementation for the broad phase contact detection, also aids the performance and the flexibility of the proposed and implemented method.

The method is shown to be convergent with respect to particle resolution, both for individual particle collisions and also for laboratory scale particle systems. The HPC implementation is proven to be highly efficient, where, for instance, a one second simulation of one million non-convex particles is simulated within an hour on a single GPU. By the effective filtering of triangles in the narrow contact phase, near linear scaling can be achieved with regards to particle resolution.

Keywords: Discrete Element Method, Polyhedral intersection, GPU, HPC, Particulate systems, Non-convex particles.

Acknowledgements

Foremost I want to acknowledge my supervisor Klas Jareteg for always taking time for discussions. Your input on everything between physics, algorithms and report writing have been highly valuable and appreciated during this thesis work. I also want to acknowledge Johannes Quist, the project lead of the DEM research group at FCC, whose insights into DEM modelling have been of great value. Also in general has the DEM research team at FCC been of great support, I am excited to continue working with all of you. Further I want to thank my examiner Anders Logg for giving valuable feedback and spending your time on this thesis.

I want to acknowledge FCC in general, and in particular department head Fredrik Edelvik, for letting me do my thesis on an interesting yet relevant topic. I also want to thank FCC for the great technical support which has enabled me to perform my thesis work uninterrupted even through the current exceptional circumstances.

Finally, marking the end of my 5 years of studies, I want to thank my family and friends for supporting me throughout these years.

This work was carried out within the framework of the Centre for Additive Manufacturing - Metal (CAM2), supported in part by the InfraSweden2030 project DigiRoad, and funded in part by the Swedish Government Agency for Innovation Systems, VINNOVA.

Adam Bilock, Gothenburg, June 2020

Contents

Abbreviations

BFS Breadth First Search.
BVH Bounding Volume Hierarchy.

CUDA Compute Unified Device Architecture.

DEM Discrete Element Method.
DFS Depth First Search.

FMA Floating point Multiply Add.
FPGA Field Programming Gate Array.

GPGPU General Purpose computing on Graphics Processing Unit.
GPU Graphics Processing Unit.

HM+D Hertz-Mindlin-Deresiewicz.

ILP Instruction Level Parallelism.
IPS Industrial Path Solutions.

NDEM Nonsmooth Discrete Element Method.

SM Streaming Muliprocessor.
SP Streaming Processor.

List of Figures

List of Tables

1
Introduction

There is a large demand for understanding the nature of granular materials from both science and industry. An increasingly popular method to study these systems is the Discrete Element Method (DEM). By simulating the particles as distinct elements, DEM can achieve a level of detail and generality not present in alternative methods for granular materials. This is opposed to continuum methods where instead the system is treated as a continuous material. DEM first originated in 1979 from implementations by Cundall and Strack[1], but due to the computational burden of resolving each distinct particle it is only in recent years the method been commonly adopted.

To lower the computational cost a common approach of DEM is to approximate all particles as spheres [2]. However, it is evident that spheres cannot represent the often varying particle shapes present in granular materials. The most popular remedy of this problem is to use compounds of spheres (here on referred to as multispheres)[3]. The simplicity of contact detection for spheres is preserved with the multispheres and, in addition, the ability to capture the shape of granular material is improved in comparison to spheres [4]. However, a deficiency of the multisphere approach is the lack of representation of sharp edges, often present in real particles.

More importantly all the force models in use for multispheres in the open literature are fundamentally flawed. This is due to that the exact intersection of two multispheres cannot efficiently be resolved and, consequently, the acting force on the particles are naively accumulated from all compounded spheres [5]. This deficiency of the force model for multispheres results in non-converging behaviour of the method, where on one hand more compound spheres are needed for an accurate shape representation and on the other hand more compound spheres makes the force model ever more inaccurate [6].

Henceforth, to attain an accurate representation of granular material a better model of the particles is needed. One candidate is polyhedrons, which by its flexible representation can be made to accurately model most realistic shapes. Furthermore, such a formulation can also, in contrast to purely algebraic models, capture the sharp edges of e.g rock material often studied with DEM. However, there is a significant computational burden of polyhedral DEM. To lower the computational burden some early efforts[7] simplified the problem by using the indentation depth on a feature-by-feature manner, meaning that the indentation depth is computed for each face-face, edge-face and vertex-face of the polyhedron. With this approach the same issue as with the multisphere appears, i.e. the force model is not decoupled from the geometric mod-

elling[6] and the only advantage over the multispheres is the more flexible geometry of polyhedrons. However, the problem with a non-converging method can be resolved with polyhedrons, namely by ensuring that the exact volumetric overlap of the particles are computed. In *Polyhedral particles for the discrete element method* by B. Nassauer et. al. [8] this approach was explored by presenting a general framework for the method. This was followed by another article [9], where a more advanced force model was derived from Hertz contact law, and was shown to agree well with FEM simulations. However, only systems of at most 500 particles were studied. In the work of N. Govender et.al. [10, 11, 12, 13] it was shown how a GPU polyhedral DEM utilizing volumetric overlaps can be effectively applied to large scale industrial usage. In their work, for instance a one second simulation of one million non-convex particles were simulated overnight on a single desktop computer.

Besides the open literature, efforts in the realm of polyhedral DEM have also been made in commercially available DEM software. Most notable in the software `Rocky DEM`, supporting multi-GPU simulations and a large extent of features. However, their work is mostly not relevant for this thesis, since neither their computational method nor other displays of their accuracy are published, and it is thus impossible to do any comparisons. Nevertheless, according to N. Govender et. al[14] `Rocky DEM` neither resolve the exact volumetric overlaps nor attain the same level of throughput as Govender et.al. have displayed, but no further details were given about the simulation case and thus the results are difficult to relate to in a scientific context. Recently `Becker 3D` introduced a polyhedral DEM GPU solver, but again no reports of their method, accuracy or other capabilities makes a direct comparison likewise impossible.

Since 2016 the Fraunhofer-Chalmers Centre (FCC) has developed a DEM framework. The current framework consists of a state-of-art GPU implementation of DEM for spherical particles and a CPU implementation of multisphere DEM. The spherical GPU solver currently allows for more than 50 million polydisperse spherical particles. Furthermore, the performance of the CPU implementation of the multisphere solver has shown to be a limiting factor for industrial scale studies of e.g. infrastructure and rock material handling simulations. In addition, the aforementioned issues[6] regarding the force models of multisphere DEM has also been exposed in research work conducted by the FCC DEM group. As a consequence of the performance and model limitations, the FCC DEM group have a need for a performant GPU DEM solver that can represent non-spherical particles and also to evaluate the polyhedral representation of particles. With the developed competence in HPC based on GPUs, a polyhedron DEM framework is a feasible next step in state-of-art software for particulate systems.

1.1 Project scope limitations and aim

The project work was conducted at FCC in the DEM research group. The purpose of the thesis work was *to implement and verify a state-of-the-art polyhedron DEM GPU code with performance equivalent or better than what has been reported in the literature*. To acquire this goal, the project work was delimited in some aspects:

- The project was implemented directly in the existing DEM framework, minimiz-

ing the efforts required for setting up the software infrastructure.
- The polyhedron particles were provided at the start of the work and no further studies were conducted on the triangulation process of the model particles themselves.
- The force model was not the primary concern of the thesis and tentatively models from the literature were to be used.

1.2 Research questions

In this work several research questions revolving the algorithmic aspects are to be answered:
- What is the current state-of-art in the open literature as regards to DEM simulations of polyhedron particles?
- Compared to the spherical GPU solver at FCC: What is the performance characteristics for the polyhedron code?
- Can a linear scaling with the number of particles be achieved on a state-of-art GPU?
- How does the computational time of the polyhedral code scale with the resolution of the triangulation of the particles?

Furthermore, research questions regarding the use of the polyhedral particle representations in DEM were formulated:
- Can convergence in physical behavior of the particles be shown with regards to the particle resolution?
- How finely resolved particles are required to achieve convergence in terms of their physical behaviour?
- Can the method capture the range of repose angles found in experiments in a calibration rig?
- Finally, and more generally: what are the benefits respectively disadvantage of a polyhedral particle representation?

1.3 Outline of thesis

Section 2 introduces GPU programming, the implications of which will determine many later decisions in the used methods. Section 3 gives an overview of DEM. Then, the following sections will go into details of the required components of polyhedral DEM. First the force models are introduced in section 4, followed by the contact detection detailed in section 5. The remaining sections will analyze the method, first the method is verified in section 6 whereafter the performance of the solver is analyzed in section 7. Finally the thesis is summarized in section 8.

2

General-Purpose Computing on Graphics Processing Units (GPGPU)

With the advent of the power wall[15], meaning that the clock rate of single core architectures could not be further increased due to thermodynamical laws, progress in computational power have been shifted towards parallel computation on a single chip. In the forefront of this shift lies general-purpose computing on graphics processing units (GPGPU). With the initial purpose of rendering 3D graphics, the hardware in GPUs have due to the parallel nature of rendering been developed and optimized for highly parallel computation. The underlying computational units of the GPU follows the stream processing paradigm which by its constrained computational model can attain a much higher throughput per power than the traditional CPU architecture. Some aspects of both the hardware architecture and corresponding SIMT (Single Instruction Multiple Threads) programming model are important for some of the algorithmic choices of this thesis and, thus, such aspects are carefully described in this section. //

2.1 Hardware architecture

The GPU is a so called co-processor and hence is its execution entirely separated from the execution of the host (meaning the CPU). In practice this means that the GPU in a broad perspective largely requires the same elements as the CPU. This means that it has its own global memory, it has a similar cache hierarchy and the execution of the underlying computational units is completely organized by the GPU itself. While the hierarchy of the GPU is similar to the CPU, the components of this hierarchy have largely been altered to accommodate the massive parallelism on offer [15].

Starting at the lowest level of the architecture, the underlying computational units of the GPU follows the aforementioned streaming process paradigm (these computational units are often referred to as Streaming Processors, SP). In the Nvidia chips a SP is represented by 32 floating point units and 32 integer units. By strictly following the streaming process paradigm all 32 units must execute the same instruction. By this restriction a much higher number of computational units can reside on a single chip opposed to a CPU. The impacts of this limitation on the programming model will be further considered later.

Each SP does merely consist of the actual arithmetic logic units (ALU) and to actually execute instructions and interact with memory each SP resides within a Streaming

Figure 2.1: The Nvidia Volta streaming multiprocessor (SM). Most required hardware for computation resides within the SM. Note that each marked computational unit in the figure, e.g. the marked FP64 units, is here a streaming processor and consists of 32 ALUs in total. Also noteworthy is the tensor cores which takes up nearly half of the space for computational units and by its restricted execution of half-precision 4×4 matrix FMA (Fused Multiply Add) operations cannot be readily utilized for many applications. [16]

Multiprocessor (SM). These units organize the computation of the SPs, where each SM consist of many SPs. Figure 2.1 shows the SM for the Nvidia Volta architecture[16]. In this architecture the different ALUs of the SP have been separated such that each SP consist of one FP32 unit, one integer unit and one FP64 unit. Further, each SM has its own L1 cache, both instruction and data, and also importantly it has access to the so called shared memory. The shared memory has access times comparable to the L1 cache, but can be manually controlled by the programmer. Moreover in the Volta architecture[16], and also the succeeding Turing architecture[17] each SM has access to 8 tensor cores. These units can execute 4×4 matrix FMA (Fused Multiply Add) operations, albeit only at half precision (meaning a 16-bit representation). Finally each GPU consist of many SMs (e.g. the Volta architectures have 80 SMs per GPU) and on this level there is also a L2 cache which interacts with the global memory. An important limitation of the GPU is that the synchronization of the SMs is not especially effective, often the execution between the different SMs rather is synchronized at the host and

usage of for example global atomics should be avoided.

A note regarding the GPU hardware is that in comparison with the CPU the amount of memory (throughout the hierarchy) available per thread is significant lower, which in practice means that significant care is required such that consecutive threads access memory linearly. Finally since the GPU is a co-processor, care must be taken to avoid transferring to much data between the GPU and the host, which for the implementation of this thesis is avoided by having all key computational steps performed by the GPU.

2.2 Programming model

In this work the CUDA programming model[18] will be used as a extension of the programming language C++. Some aspects of this programming model will be of importance for the algorithmic implementations. The basic model of CUDA consists of a host program that launches GPU kernels. A kernel consist of many threads executing the same program. The host execution primarily serves as a global synchronization point of the GPU and managing the control flow of the application. This role of the host is partly induced by the lack of effective global synchronization of the different SMs as previously discussed, but also due to that most control flow tasks are single threaded and thus more suitable for the CPU.

So far the programming model follows the streaming process paradigm. However, CUDA also exposes a lower granularity of the thread organization which maps to the aforementioned hardware hierarchy. Foremost the threads are divided into blocks. Each thread block can only be executed on a single SM, where the threads in a block can be synchronized within the kernel and all have access to the same shared memory. Utilizing the shared memory is essential to attain effective kernels, however it is severely limited in size (between 46-96kB[16, 17]). Furthermore, each thread block consist of several thread warps, where a thread warp resides in a single SP. The threads in a warp have faster synchronization and can to certain extent communicate by and share registers. A significant limitation of the programming model is that all threads in a warp need to execute the same instruction, this is imposed by the underlying computational units following the streaming process paradigm. Thus can the performance significantly degrade if the execution of the threads in a single warp diverges. This limitation means a significant challenge for the programmer to ensure that control flow within the kernels minimize the number of branches. This limitation does in practice mean that often simpler more straightforward algorithms should be preferred opposed to what typically is the most efficient solution on the CPU [15]. It should be mentioned that this limitation of thread divergence have with the newer architectures, Volta[16] and Turing[17], been slightly relaxed, but diverging control flow should still be minimized.

2.3 Implementation details

In the implementation of the methods presented in this thesis, the impact of low level technical details on the GPU have shown to give significant change in performance. Due to the large scope of the thesis, not all such details can be enclosed, rather this section will highlight some key programming patterns which has enabled increase of the performance.

A recurring pattern is to always minimize the data usage. This is two-fold, it is important both to minimize the actual data footprint, but also to reduce the data flow of the application. This affects both the choices of methods, and the implementation details such as effectively packing the data, using custom memory allocators and ensuring that the data gets properly cached during computations in the memory hierarchy. This, at times, could be taken to the extremes, where it was beneficial to reorder data before key computational kernels in order to improve the cache hit rate during the actual computation. Similarly, also large performance benefits could be achieved by reordering data to minimize the aforementioned thread divergence.

The generated code from the compiler is often non-optimal [19, 20] for CUDA, this is a result of both choices in the design of the C++ programming language and, in some cases, limited compiler support for CUDA. Thus, for many of the key computational kernels in the implementation, the code was analyzed on an assembly instruction level. Such an analysis often disclosed how small, almost trivial, changes to the C++ code could lead to significant changes in the generated machine code and ultimately the performance of the solver. One such example is aliasing [21], where the C++ language does not guarantee that two pointers of the same type do not point to the same memory location. If aliasing is not remedied it can in certain instances lead to a significant higher memory throughput usage since there is no guarantee that the underlying data have not changed and thus data has to be reloaded from the global memory at each usage. Another example, is that small changes in the C++ code could hinder the compiler to vectorize loops, which results in worse instruction level parallelism (ILP), in turn also significantly degrading the performance.

To quickly iterate the fine tuning of the low level details, most key computational kernels were micro-benchmarked, meaning that the performance of single kernels were measured and the process automated in the build system. Moreover, also profiling tools such as NVIDIA Nsight Compute/Systems [22, 23] were frequently used, which allowed for profiling both on a system level (Nsight Systems) but also at an assembly instruction level (Nsight Compute).

3

Overview of the Discrete Element Method

In DEM, the particles are simulated as rigid bodies where the state of each particle is individually tracked and evolve over time due to various interactions with e.g. other particles or materials. The dynamics of the particles are determined by Newton's second law, namely,

$$\begin{aligned} m_i \frac{\mathrm{d}^2 \mathbf{x}_i}{\mathrm{d}t^2} &= \mathbf{F}_i \\ I_i \frac{\mathrm{d}^2 \Theta_i}{\mathrm{d}t^2} &= \mathbf{T}_i \end{aligned} \tag{3.1}$$

here m_i are mass, $\mathbf{x}_i$ is position of the centre of mass, $\mathbf{F}_i$ is the acting force, I_i is the inertia, Θ_i is the orientation and $\mathbf{T}_i$ is the acting torque, respectively on particle i. Commonly eq. 3.1 are integrated by explicit methods. However, also implicit versions of DEM exist, and are then often referred to as nonsmooth DEM (NDEM)[24] which corrects for the fact that the forces and torques in eq. 3.1 cannot be expressed as a direct mapping from the particle state in the previous time step. NDEM can for certain applications compensate the increased computational burden induced by an implicit method by allowing for the use of larger time steps. However, for most applications, explicit formulations are preferred since then eq. 3.1 can be solved for each particle independently, whereas for the implicit counterpart the resulting matrix systems from eq. 3.1 are significantly more evolved to solve. Henceforth, in the remainder of this

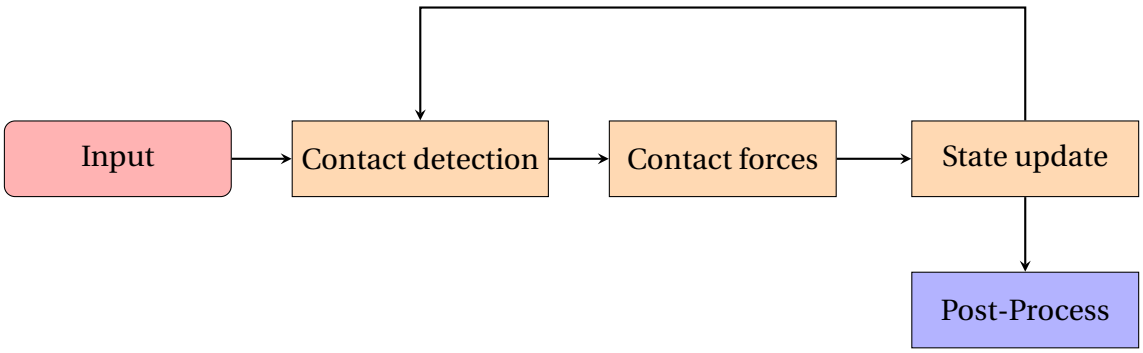


Figure 3.1: Overview of the control flow of the discrete element method. Given input data the method iterates each time step by; first resolving the contacts of the particles, then evaluating the contact forces and finally the state of the particles are updated according to eq. 3.1.

work only explicit DEM is considered.

The control flow of a time step in an explicit formulation of DEM can be seen in figure 3.1. Here contact detection refers to finding the interacting particles, and if present also the interactions of the particles with other bodies (e.g. static world geometries). Note that each particle contact pair is resolved independently. The contact detection is typically performed in several phases; a broad phase, a narrow phase and finally resolving the geometric overlaps. The broad phase contact detection often relies on bounding volumes of the particles (e.g. axis-aligned boxes) which aims to efficiently, but approximately, filter out all potential pairs. Then for the case of complex particle shapes a narrow phase contact detection is often performed, which further filters out potential pairs but in some cases also individual components of the complex shapes (e.g. triangles or spheres). For simple spheres the narrow phase is often not required. Finally the actual overlap of the particles are resolved, the exact details of this most detailed phase depends on the particle representation and the force model used. The contact detection methods of this work are further discussed in section 5.

With the contacts resolved, the next step consists of computing the interacting forces to give the right hand side of eq. 3.1. In this work only interactions resulting from direct geometric contact are consider, i.e. the particle has no long-range forces. The formulation of the contact forces in this work can be seen in section 4. Finally the state of the particles are updated by integrating eq. 3.1 which is further detailed in section 3.4. Now the simulations are performed over time by iterating over the time steps.

3.1 Limitations

With the explicit DEM formulation there are certain key assumptions and limitations to consider. One of the main assumptions is that the particle geometry is assumed to be rigid, meaning that there is no deformation or plasticity in the particles. Still, the rigid geometries of the particles are allowed to overlap, but this is only valid if the overlap of the particles are assumed to be small enough such that effects from deformation or other plasticity are small. This limitation has the negative effect of restricting the method to small time steps such that these small overlaps are attained at each step [2].

3.2 Discrete element representations

As discussed in the introduction (see section 1) there are several different representations of the discrete elements in DEM. The most common shape is simple spheres, which largely is due to their computational advantages. However, as many studies have shown the effect of the shape of particles on the systems can be profound [12, 4]. Henceforth, several methods to represent irregularly shaped particles have been developed. The most common method is the multisphere representation, where each particle is represented by several clumped spheres. Moreover, there are several attempts of different analytical representations e.g. ellipsoids. Further, most relevant for this

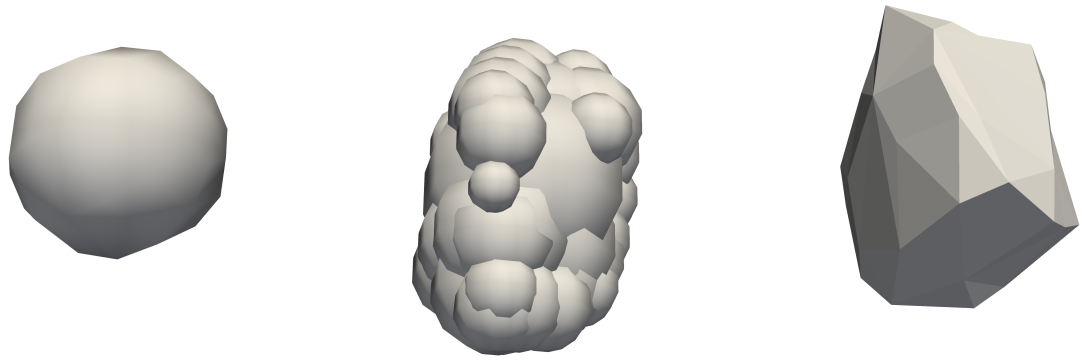


Figure 3.2: The three most common particle representations in DEM. The simple sphere, multispheres and polyhedrons (left to right).

thesis is the polyhedral representations. Figure 3.2 shows three most common particle representations in DEM, a simple sphere, multisphere and polyhedron. Note that for all approaches there are typically only a few particle models per population, where often only the scale of each model can be changed for each particle, this is due the high memory requirements of having unique models for each particle. Figure 3.3 gives a overview of the associated computational cost and physical fidelity of the different particle resolutions.

Spheres have commonly been used in DEM. Largely the reason for this has been the computational advantage of such a simple shape. However, it is evident that spheres cannot represent irregular shaped particles often found in granular materials. While this can be to some extent compensated for by using non-uniform rolling friction of the spheres[4] this is accompanied with significant tuning of the parameters which significantly reduce the usability, and ultimately the validity, of the method. The scale of the current state-of-art spherical DEM solvers is in the order of tenths of millions of polydisperse spheres [25]. It has also been displayed in the literature how the usage of GPUs can significantly speed up the simulation time, where speed up ratios between 60-100 times single threaded CPU implementations have been shown [25].

Multispheres consist of several clumped sub-spheres as displayed in figure 3.2. The sub-spheres of one multisphere typically remains fixed in the particle body frame. The ability to model irregularly shaped particles is improved in comparison to spheres and it has been shown how the clumped representation can to some extent give predictive results with respect to particle shape[4]. The ease of contact detection also remains with multispheres where only slight modifications are required opposed to spherical DEM. However, a significant issue arise by its geometric modelling in the force model. Namely the contact detection of each sub-sphere is completely independent of the other sub-spheres in the particle. Meaning that each sub-sphere pair of one particle contact is modelled as separate contact forces. This issue is illustrated in figure

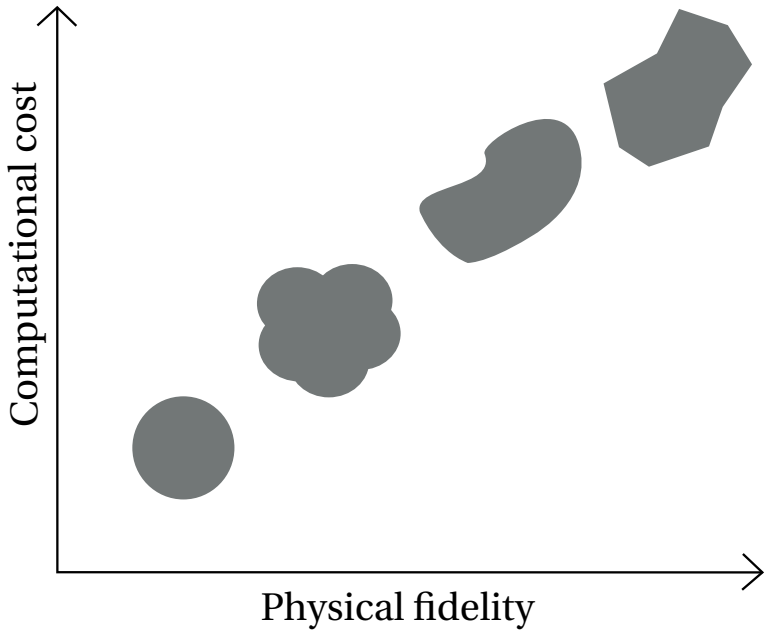


Figure 3.3: Overview of the common particle representations in DEM. The scales gives a outline of the physical fidelity and the associated computational cost for the different representations. From left (bottom) to right (top) there is simple sphere, multisphere, analytical and polyhedral particle representations.

3.4, where one particle contact is represented by 8 different sphere-sphere contacts, all these 8 sphere-sphere contacts will naively be accumulated and give a significant higher contact force compared to if the actual overlap volume could be resolved. It has been shown in the literature how this severely distort the contact forces for multisphere particles [5]. Now, since the only approach to improve the geometric modelling of the multispheres is to increase the number of subspheres this leads to a non-converging method with respect to particle resolution [6]. This issue can be illustrated by, for instance removing the smallest two spheres in figure 3.4, then the resulting contact force of the collision in figure 3.4 will be significantly lower.

Polyhedrons have largely been avoided in DEM due to the high computational demand of resolving the contacts. To reduce the computational cost the first efforts with regards to polyhedral DEM used contact forces by an feature-by-feature manner, meaning that each face-face, face-edge and edge-edge contact were independently resolved. Notable work utilizing this approach is found in Govender et. al. [7, 26] where primarily the preeminent advantage of utilizing the GPU also was shown for polyhedral DEM where millions of particles could be simulated on a single GPU. However, this contact detection scheme similarly to the multisphere approach can severely distort the contact forces and again the force model is not decoupled from the geometric modelling [6]. This problem can be avoided for polyhedrons by instead resolving the exact volumetric overlaps of the particles. Notable work using this contact detection scheme is seen in Nassauer et. al. in [8], where a general framework for the method was developed, and also in [9] where a more advanced force model was derived which attempts to take the contact curvature into account which results in good agreement

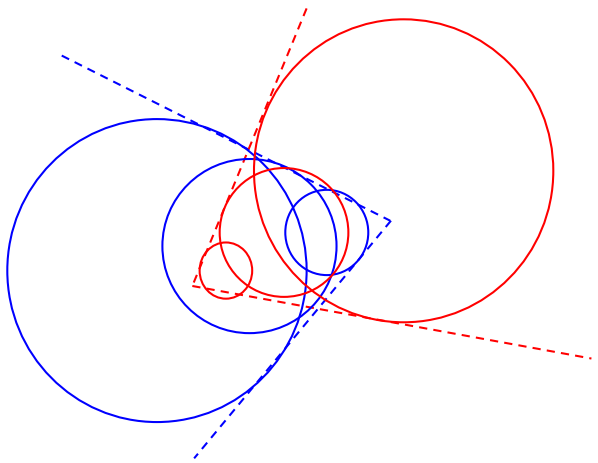


Figure 3.4: The figure shows collision of two objects. The actual objects are represented as the dotted lines, the spheres with the same colour as the lines are part of the multisphere representation of that object. For this case, the multisphere representation would give very strong contact force, since all individual sphere-sphere contacts between the blue and red spheres will be counted as individual forces.

with FEM simulations. However, in [8, 9] the system sizes are restricted to 500 particles and the solver was not adopted for non-convex particles. Instead Govender. et. al. have in recent years[11, 12, 13] adapted the initial solver from [7, 26] to instead resolve the exact volumetric overlaps of the particles. In their work also non-convex particles could be simulated by decomposing the particles into convex parts. Govender et. al can achieve this physical fidelity while still being capable of simulating large system sizes, for instance a one second simulation of one million non-convex particles could be simulated overnight. It is noteworthy that in the work of Govender et. al. a slightly simpler elastic force model opposed to the one proposed by Nassauer et. al. was used which to not the same extent corrects the force model to the contact curvature.

3.3 World geometries

For DEM to be usable, the particles also need to interact with a surrounding environment. Rigid world geometries is the most common such an environment. Several different methods exist to model the world geometries; purely as analytical surfaces (e.g. planes and cylinders), as several rigid clumped particles or as triangle meshes. The former alternative, with purely analytical surfaces, can be the most straightforward when only a few different world geometries are considered, but this approach also severely restricts the usage of the method. In contrast, more general approaches such as interactions with CAD surfaces is significantly more evolved. The second approach, rigid clumped particles, are a popular method for polyhedral DEM[8, 9], since then the contact detection for the world geometries and particle interactions are completely equivalent. For certain instances the conversion between triangle mesh to clumped polyhedron particles can be straightforward, however for complicated surfaces the con-

version can be much more complicated and hence is this option not especially user friendly. Only recently in literature on polyhedral DEM have the case of world geometries represented by triangle meshes been considered [13]. Using a triangle mesh to represent the world geometry is the most flexible approach since they can both be created from point clouds of scanned objects or be generated as computational meshes from analytical surfaces. For the case of polyhedral DEM, the contact forces between particle and triangle mesh can remain the same as for particle-particle interactions, see section 4. Further, there is also only slight changes required for the contact detection in comparison with the particle interactions, which is further discussed in section 5.6.

3.4 Integration

In this work explicit integration is used for solving the ODE system of eq. 3.1 and as previously mentioned is the standard approach in DEM. A common method first proposed by Cundall and Strack in 1979[1] for the translational integration is the Velocity Verlet algorithm. The translational integration then goes as following:

1. Update velocity, $\mathbf{v}_i(t+\frac{1}{2}\Delta t) = \mathbf{v}_i(t) + \frac{1}{2}\mathbf{a}_i(t)\Delta t$
2. Update position, $\mathbf{x}_i(t+\Delta t) = \mathbf{x}_i(t) + \mathbf{v}_i(t+\frac{1}{2}\Delta t)\Delta t$
3. Compute force, $\mathbf{f}_i$, based on $\mathbf{x}_i(t+\Delta t)$ and $\mathbf{v}_i(t+\frac{1}{2}\Delta t)$, update acceleration, $\mathbf{a}_i(t+\Delta t) = \mathbf{f}_i / m_i$
4. Update velocity, $\mathbf{v}(t+\Delta t) = \mathbf{v}(t+\frac{1}{2}\Delta t) + \frac{1}{2}\mathbf{a}_i(t+\Delta t)\Delta t$

where i refers to the particle $i = 1, ..., n$. Note that the force computation is performed at positions and velocities that are a half-step apart, which for the viscous forces of DEM are deemed acceptable through empirical evidence[27]. For the orientation explicit forward Euler integration is used.

4

Contact Forces for Irregular Shaped Particles

This section will present the contact forces used in this study, i.e. the expression needed to evaluate the right hand side of eq. 3.1. The model will be derived from the well studied Hertz-Mindlin-Deresiewicz (HM+D) model for spheres. Hence will the spherical model first be described and then it will be shown how it can be adapted to arbitrary shapes as first shown in[9] for the regular Hertz contact law for spheres.

Note that in an earlier phase of this work the model proposed by [9] was intended to be used. However, as further detailed in section 6.3 the desired behaviour of particle systems could not be achieved with this model, hence in section 4.2 an additional tangential spring stiffness are introduced to address this issue.

4.1 HM+D model for spheres

The HM+D model has a normal elastic force which corresponds to the Hertz contact law for spheres[28],

$$F_{n,e} = \frac{4}{3} E^* R^{1/2} \delta^{3/2} \tag{4.1}$$

where E^* refers to the effective modulus, R^* is the effective radius of the spheres and δ is the indentation depth. The effective modulus is defined as,

$$\frac{1}{E^*} = \frac{1-\nu_1^2}{E_1} + \frac{1-\nu_2^2}{E_2} \tag{4.2}$$

where ν_i is the Poisson's ratio and E_i is the Young's modulus for each sphere $i = 1,2$. The effective radius is defined as,

$$\frac{1}{R} = \frac{1}{R_1} + \frac{1}{R_2} \tag{4.3}$$

where R_i is the radius of the sphere $i = 1,2$. Note how the intersection with a sphere and a plane can be defined by letting $R_2 \to \infty$ and consequently $R = R_1$. Finally, the indentation depth for sphere-sphere contacts can be calculated as,

$$\delta = \begin{cases} R_1 + R_2 - |c_2 - c_1| & \text{if } |c_2 - c_1| < R_1 + R_2 \\ 0 & \text{otherwise} \end{cases} \tag{4.4}$$

where c_i is the center of each sphere $i = 1,2$. Further is the scalar dissipative normal force given by,

$$F_{n,d} = 2\gamma\sqrt{m^* k_n}\upsilon_n \tag{4.5}$$

where γ is a damping coefficient, m^* is the effective mass of the particles, k_n is the normal spring stiffness and υ_n is the relative velocity in the normal direction. The effective mass is given by,

$$m = \frac{m_1 m_2}{m_1 + m_2} \tag{4.6}$$

and the normal spring stiffness is given by,

$$k_n = 2E^*\sqrt{R\delta}. \tag{4.7}$$

In the HM+D model a tangential spring force is derived from the *no-slip* theory of Mindlin [29]. The scalar component of this force is given by,

$$F_{t,e}^n = F_{t,e}^{n-1} + k_t^n \Delta\delta_t \qquad \text{if } \Delta F_n \geq 0 \tag{4.8}$$

$$F_{t,e}^n = F_{t,e}^{n-1}\left(\frac{k_t^n}{k_t^{n-1}}\right) + k_t^n \Delta\delta_t \qquad \text{otherwise} \tag{4.9}$$

where k_t is the tangential spring stiffness, n refers to the current time step and $\Delta\delta_t$ is the tangential displacement at the current time step. The tangential spring stiffness is given by the following expression,

$$k_t = 8G^*\sqrt{R\delta}, \tag{4.10}$$

where G^* is the effective shear modulus, given by,

$$\frac{1}{G^*} = \frac{1-\upsilon_1^2}{G_1} + \frac{1-\upsilon_2^2}{G_2}. \tag{4.11}$$

The tangential force also has a dissipative component, giving the final expression for the scalar tangential force as,

$$F_t = F_{t,e}^n + 2\gamma\sqrt{mk_t}\upsilon_t \qquad \text{if } F_t < \mu F_n \tag{4.12}$$

$$F_t = \mu F_n \qquad \text{otherwise} \tag{4.13}$$

where μ is a friction coefficient.

4.2 HM+D model for irregular shaped particles

Now following the work of Nassauer et. al [9] a contact model for arbitrary shaped particles can be derived. This is done by noting that the volume of a sphere-sphere intersection can be approximated as,

$$V \approx \pi\delta^2 R \tag{4.14}$$

if one assumes $\delta \ll R$. Similarly for a sphere-plane intersection the volume is given by,

$$V = \frac{1}{3}\pi\delta^2(3R - d) \tag{4.15}$$

which again can be approximated according to equation 4.14 when $\delta \ll R$. The assumption of, $\delta \ll R$, i.e. small overlaps relative to the particle size is not a restrictive assumption, since as mentioned in section 3 this is already assumed in DEM modelling. Now inserting this expression into the HM+D model for spheres the following scalar normal force for arbitrary shaped particles is obtained which instead of the radius is based on volume,

$$F_n = \frac{4}{3\sqrt{\pi}}E^*\sqrt{V\delta} + 2\gamma\sqrt{m^* k_n}\upsilon_n, \tag{4.16}$$

but now the normal spring stiffness is defined as,

$$k_n = 2E^*\sqrt{\frac{V}{\pi\delta}}. \tag{4.17}$$

Further, the expressions in eq. 4.8 and eq. 4.12 for the tangential force remains the same, but for an irregular shaped particle the tangential spring stiffness can instead be defined as,

$$k_t = 8G^*\sqrt{\frac{V}{\pi\delta}}. \tag{4.18}$$

Again following the work of Nassauer et. al. [9] the direction of the normal force is calculated by integrating over the surface normal of one of the particles,

$$\mathbf{n} = \frac{\int_{A_1}\mathbf{n}_s\,\mathrm{d}s}{\left|\int_{A_1}\mathbf{n}_s\,\mathrm{d}s\right|} = \frac{\sum_i A_1^i\mathbf{n}_s^i}{\sum_i A_1^i} \tag{4.19}$$

where $A_1 = \sum_i A_1^i$ is the area belonging to one of the particles in the intersection polyhedral and $\mathbf{n_s}$ is the surface normal. Further the indentation depth, δ, is defined as the maximum cross sectional length of the intersection in the normal direction. Finally the contact point of the interaction is defined as the center of mass of the volumetric overlap of the particles. Note that the same contact force model can be used for particle interactions with the triangle mesh world geometry. However, it requires the restriction on the world geometry to have a closed surface and have a defined interior, note that the interior opposed to the polyhedral particles can still be an open sub-space of $\mathbb{R}^3$.

With the contact forces well defined, the right hand side of eq. 3.1 can be determined. Furthermore, as the required properties for each particle overlap are known, it now remains to compute these properties for polyhedrons, which will be considered in the next section 5.

5

Polyhedral DEM Contact Detection

Figure 5.1 shows an intersection of two particles. The derived contact forces from section 4 will need several properties of the overlapping volume from each such contact pair. Namely, the volume, V, of the overlap needs to be computed (the marked grey area), further to attain the direction of the force, $\mathbf{n}$, eq. 4.19 needs to be evaluated which are equivalent of integrating the normals along the red triangles marked in figure 5.1. Moreover, to get the indentation depth, δ, the extent of the overlapping volume needs to be evaluated along the direction $\mathbf{n}$. Also the center of mass, $\mathbf{c}$, of the overlapping volume needs to be evaluated to give a contact point of the interaction. In the following, the term mass properties will be used to refer to all the above properties of the volumetric overlap. This section will present the required theory and the method used for finding such properties for each particle contact. This section will also consider the broad phase contact detection i.e. to effectively, but approximately, filter out all contact pairs, both particle-particle and particle-triangle mesh contacts.

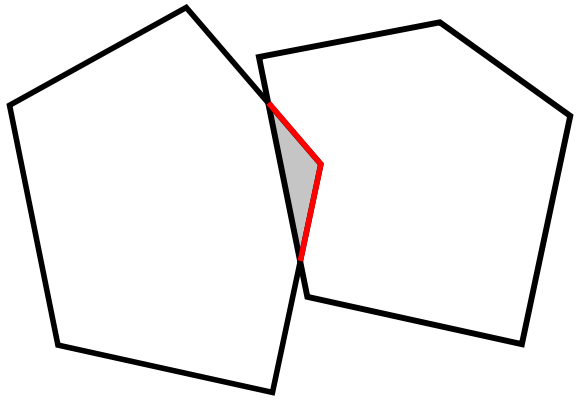


Figure 5.1: Collision of two polyhedral particles. The intersection volume is marked in grey. The surface integrated to attain the normal force direction is marked as red.

5.1 Polyhedron representations

A polyhedron is defined as a 3-dimensional solid composed of several polygons [30]. Typically, the polygons are connected at the edges. When using volumetric overlaps for DEM particles an essential restriction is that the interior of the polyhedron represents a closed subspace of $\mathbb{R}^3$. The convex polyhedron is an important special case, it can

be defined as the intersection of several half-spaces, i.e the bounded solution, $\mathbf{x}$, to the linear equation,

$$A\mathbf{x} \leq \mathbf{b} \tag{5.1}$$

where $A \in \mathbb{R}^3 \times \mathbb{R}^m$ and $\mathbf{b} \in \mathbb{R}^m$. For two convex polyhedrons represented by $A^{1,2}$ and $\mathbf{b}^{1,2}$, the intersection is defined as the bounded solution, $\mathbf{x}$, to the linear equation,

$$A'\mathbf{x} \leq \mathbf{b}' \tag{5.2}$$

where $A' = [A^1, A^2] \in \mathbb{R}^3 \times \mathbb{R}^{n+m}$ and $\mathbf{b}' = [\mathbf{b}^1, \mathbf{b}^2] \in \mathbb{R}^{n+m}$. Note how this implies that also the solution, $\mathbf{x}$, is convex. These properties of convex polyhedrons will have large implications when intersection algorithms are considered in the next section 5.2.

5.2 Intersection nodes

The intersection nodes of the volumetric overlaps needs to be computed in order to acquire the desired mass properties of each polyhedral intersection (see section 5.3 how the mass properties are computed from the intersection nodes). The intersection nodes of two general polyhedrons are defined as the extreme nodes of each convex part of the intersection. Namely, there are two types of intersection nodes (see also figure 5.2):

- Already existing nodes of one of the polyhedrons which reside in the interior of the other polyhedron.
- New nodes formed from an edge of one of the polyhedrons intersecting a triangle from the other polyhedron.

In previous work with regards to polyhedral DEM[8, 11] the intersection nodes have in essence been found from iteration over the respective features (e.g. triangles/edges or half-planes) of each polyhedron. In [8] only convex particles were considered and the half-plane representation (see eq. 5.1) was used, whereas in [11] triangulations were used. A naive pseudo algorithm which computes the intersection nodes by iterating all features for a triangulation can be seen in Algorithm 1, note that no assumptions are needed on the convexity. Given two polyhedrons, each with n features, it directly follows that this algorithm will scale as $O(n^2)$. While for the convex case there exist theoretically optimal algorithms with worst case time complexity of $O(n)$ [31, 32].

However, it is unclear if these theoretical optimal algorithms pose an actual advantage in terms of performance, since the constant factor of these theoretical optimal algorithms relative to the naive algorithm are unknown. This is especially true on the GPU, where, as outlined in section 2.2, simpler algorithms are typically preferred and the complicated control flow (e.g. one is recursive) of these theoretical optimal algorithms cast doubts how suitable they would be for a GPU implementation. Further, e.g. rock material can have many small concavities and thus it would be preferable to have algorithms capable of non-convex particles. No proven optimal algorithms for non-convex polyhedrons have been shown in the literature, and most implementations rely on convex decomposition [33]. Triangulations are even commonly decomposed into the smallest possible convex polyhedron, the tetrahedron, and the intersection for each tetrahedron is then resolved[34]. However, most literature on the subject of

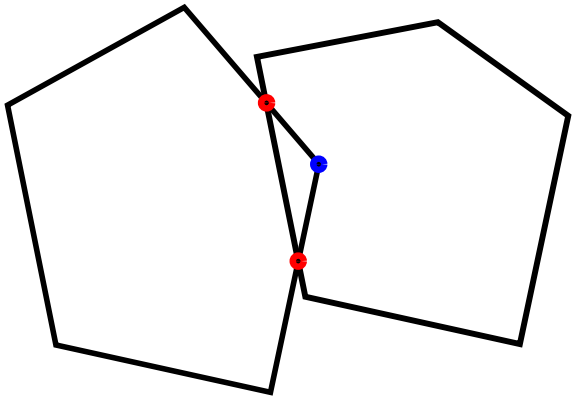


Figure 5.2: Intersection points of the polyhedral intersection is marked as dots. The blue dot shows an already existing node, the red dots are new nodes formed from triangle-edge intersections.

non-convex polyhedral intersection works under the assumption of having a few large triangulations and the computation can then be accelerated by placing each triangle or tetrahedron into some accelerated data structure such as boundary volume hierarchies or grid files.

For polyhedral DEM there is vastly different assumptions, the number of triangles per particle is expected to be low. Moreover, as previously mentioned, these particles can have many small concavities which means that doing decomposition could substantially increase the triangle count. Further, in this work with a GPU implementation, millions of intersections are assumed to be performed concurrently. With many intersections performed concurrently the memory requirements of each intersection must be very low and consequently would using advanced data structures such as the grid file or bounding volume hierarchies for each triangle not be possible. Consequently, most algorithms presented in the literature aimed to improve the scaling of polyhedral intersection are not likely beneficial for the case of a polyhedral DEM GPGPU implementation.

Instead we propose a simple heuristic algorithm which takes advantage of a fundamental assumption of soft DEM, namely that the particles are assumed to have small overlaps (see section 3 and 4). The basic idea of the algorithm is to quickly filter out triangles which are far from the overlapping volume and then using the naive algorithm 1 on the filtered triangles. Namely, the triangles is filtered by first defining a direction,

$$\hat{\mathbf{d}} = \frac{\mathbf{c}_2 - \mathbf{c}_1}{\|\mathbf{c}_2 - \mathbf{c}_1\|} \tag{5.3}$$

where $\mathbf{c}_i$, $i = 1,2$, is respective particle center. Further, the extent of each particle in that direction is expressed as,

$$e_1 = \max_{j=1,\dots,n_{nodes}} \mathbf{v}_{j,1} \cdot \hat{\mathbf{d}}; \qquad e_2 = \min_{j=1,\dots,n_{nodes}} \mathbf{v}_{j,2} \cdot \hat{\mathbf{d}} \tag{5.4}$$

where $\mathbf{v}_{j,i}$, $j = 1,\dots,n_{nodes}$, is the nodes of polyhedrons $i = 1,2$. Now one can define half-planes orthogonal to $\hat{\mathbf{d}}$ at each distance $e_{1,2}$. The half-planes is defined as the

Algorithm 1 Given two polyhedrons, P_1 and P_2 find the intersection nodes, c.

```
Let c = {} be the intersection nodes.
for each node, p, in P1 do
  if p inside P2 then
    Let c = c ∪ {p}
  end if
end for
for each node, p, in P2 do
  if p inside P1 then
    Let c = c ∪ {p}
  end if
end for
for each edge, e, in P1 do
  for each triangle, t, in P2 do
    if e intersects t at p then
      Let c = c ∪ {p}
    end if
  end for
end for
for each edge, e, in P2 do
  for each triangle, t, in P1 do
    if e intersects t at p then
      Let c = c ∪ {p}
    end if
  end for
end for
```

following,

$$h_1 = \{\mathbf{x} \in \mathbb{R}^3 | \mathbf{x} \cdot \hat{\mathbf{d}} \leq e_1\} \qquad h_2 = \{\mathbf{x} \in \mathbb{R}^3 | \mathbf{x} \cdot \hat{\mathbf{d}} \geq e_2\}. \tag{5.5}$$

Then the triangles in each particle which does not reside in the half-plane of the other particle can be removed from consideration. Also, if the two half-planes do not intersect, then the particles cannot intersect. If there exist an intersection of the half-planes, this intersection is convex, and triangles from both of the particles that reside in the intersection are guaranteed to be included. Since algorithm 1 will be applied on the filtered triangles, the last note is important, because it enables for using ray-tracing to check if nodes are inside the polyhedrons even on the filtered triangles. An illustration of the filtering is seen in figure 5.3, where the intersection of the half-planes are marked as the grey area.

This entire filtering process can clearly be performed in $O(n)$ time complexity. It also has no additional memory requirements since the half-planes are readily recomputed at each time step. For the typical use case in DEM this filtering is effective, as later seen in section 7.2.1. There is however no guarantee of its effectiveness for completely general polyhedrons, as it is easy to construct examples where no triangles at all will be excluded.

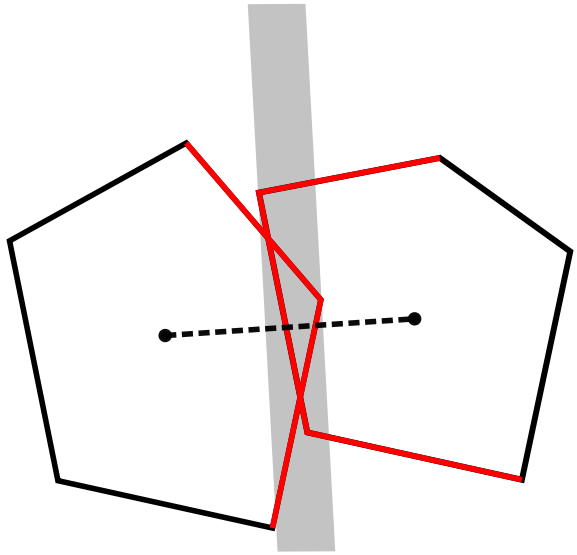


Figure 5.3: Illustration of the filtering of triangles. The grey area is the intersection of the two respective half-planes used for filtering. The red edges (in 3D triangles) are include for further computation of mass properties.

Importantly the algorithm cannot fail, this property is independent of the direction, $\hat{\mathbf{d}}$. The proof of this is trivial, since it is enough to prove that the half-plane of respective particle will in all cases cover the entire particle. This directly follows from the definition of the half-planes in eqs. 5.4 and 5.5 in combination with the fact that triangles are convex.

5.3 Intersection mass properties

The standard approach [8, 10] to evaluate mass properties of overlapping polyhedron particles in DEM consist of triangulating the intersection volume and then from the resulting triangulation calculating the desired mass properties. Further, triangulation of the volume is typically restricted to convex volumes, where e.g. convex hull algorithms or similar have been used. This method is flawed in two regards; first it is doing more work than needed and, due to the triangulation, it is restricted to convex volumes (see section 5.1). This section aims to remedy these issues by presenting an alternative method to perform these calculations.

When using triangulations to compute the mass properties one stores each constituent triangle, k, of the polyhedral intersection. Further to compute the mass properties the nodes, $\mathbf{v}_k^i$, $i = 1,2,3$, of each triangle k is needed. Then for example the volume of the triangulated intersection volume can be computed as,

$$V = \sum_{k=1}^{n} \frac{1}{6} \left| (\mathbf{v}_k^1 \times \mathbf{v}_k^2) \cdot \mathbf{v}_k^3 \right|, \tag{5.6}$$

where the other mass properties can be computed in a similar manner.

However, we propose that using an alternative data structure, based entirely on local properties of each node of the intersection, can significantly simplify the computation of the mass properties. The foundation of the data structure, introduced by W. Randolph Franklin [34], is a *cusp* defined as the following quadruple,

$$C = (\mathbf{p}, \hat{\mathbf{t}}, \hat{\mathbf{n}}, \hat{\mathbf{b}}) \tag{5.7}$$

where each vector is defined as the following:

- $\mathbf{p}$: The Cartesian coordinates of the node.
- $\hat{\mathbf{t}}$: Unit vector in the direction of an adjacent edge of the node.
- $\hat{\mathbf{n}}$: Unit vector in the direction of an adjacent face to $\hat{\mathbf{t}}$ and perpendicular to $\hat{\mathbf{t}}$.
- $\hat{\mathbf{b}}$: Unit vector in the direction of the interior of the polyhedron which is perpendicular to both $\hat{\mathbf{t}}$ and $\hat{\mathbf{n}}$.

Note also how $\hat{\mathbf{b}}$ only adds one bit of information, since $\hat{\mathbf{b}} = \pm\frac{\hat{\mathbf{n}}\times\hat{\mathbf{t}}}{\|\hat{\mathbf{t}}\times\hat{\mathbf{n}}\|}$, i.e. it only adds the sign of the vector. Now a polyhedron can be represented as an unordered list of cusps, $L = \{C_i\}$, $i = 1, .., m$ where it is noteworthy that the only constraints on the polyhedron is that it is finite and that the interior of the polyhedron is a closed subspace of $\mathbb{R}^3$. Each cusp can be seen as a signed simplex defined by the points,

$$\left(\mathbf{p}_i,\ \mathbf{p}_i - \hat{\mathbf{t}}_i\cdot\mathbf{p}_i\hat{\mathbf{t}}_i,\ \mathbf{p}_i - \hat{\mathbf{t}}_i\cdot\mathbf{p}_i\hat{\mathbf{t}}_i - \hat{\mathbf{n}}_i\cdot\mathbf{p}_i\hat{\mathbf{n}}_i,\ \mathcal{O} = \mathbf{p}_i - \hat{\mathbf{t}}_i\cdot\mathbf{p}_i\hat{\mathbf{t}}_i - \hat{\mathbf{n}}_i\cdot\mathbf{p}_i\hat{\mathbf{n}}_i - \hat{\mathbf{b}}_i\cdot\mathbf{p}_i\hat{\mathbf{b}}_i\right) \tag{5.8}$$

where the last equality trivially is seen by noting that $(\hat{\mathbf{t}}_i, \hat{\mathbf{n}}_i, \hat{\mathbf{v}}_i)$ defines an orthogonal coordinate system of all 3 dimensions.

Now to compute the area, A, volume, V , and center of mass, $\mathbf{c}$, of the polyhedron the properties are accumulated from each signed simplex independently:

$$A = \sum_{i=1}^{m} A_i = \sum_{i=1}^{m} \frac{1}{2}\hat{\mathbf{t}}_i\cdot\mathbf{p}_i\ \hat{\mathbf{n}}_i\cdot\mathbf{p}_i \tag{5.9}$$

$$V = \sum_{i=1}^{m} V_i = \sum_{i=1}^{m} -\frac{1}{6}\hat{\mathbf{t}}_i\cdot\mathbf{p}_i\ \hat{\mathbf{n}}_i\cdot\mathbf{p}_i\ \hat{\mathbf{b}}_i\cdot\mathbf{p}_i \tag{5.10}$$

$$\mathbf{c} = \frac{1}{V}\sum_{i=1}^{m} V_i\mathbf{c}_i = \frac{1}{V}\sum_{i=1}^{m}\frac{V_i}{4}(3\mathbf{p}_i + 2\hat{\mathbf{t}}_i\cdot\mathbf{p}_i\ \hat{\mathbf{t}}_i + \hat{\mathbf{n}}_i\cdot\mathbf{p}_i\ \hat{\mathbf{n}}_i) \tag{5.11}$$

For given input data, the cusp model is inferior to the regular triangle representation in terms of number of operations required for the mass property computation. Since for a polyhedron of n triangles the number of cusps required to represent this polyhedron is $m = 6n$. On a NVIDIA GPU the dot product requires 3 floating point operations (2 fused-multiply add and 1 multiplication) and the cross product 6 floating point operations (3 fused-multiply add respective multiplication). Now to compute the volume of the polyhedron the total number of floating point operations (FP) for respective method is,

$$\#\,\text{FP} \sim 9n \text{ for triangle representation}$$
$$\#\,\text{FP} \sim 11m = 66n \text{ for cusp representation.}$$

While this ratio can be made more favorable to the cusp model in the case when all three properties are computed at once, it nevertheless requires substantially more floating point operations.

However, for the case of polyhedral intersection there is a substantial advantage of the cusp representation. Namely that no triangulation of the intersection volume is required since the local properties of each cusp can be determined at each intersection point. Moreover, since the cusp model also works on completely general polyhedrons no convex decomposition of the particle models is needed. This representation also fits the streaming processors of the GPU especially well, since both the triangulation and the decomposition require more complicated control flows and hence runs a greater risk of thread divergence.

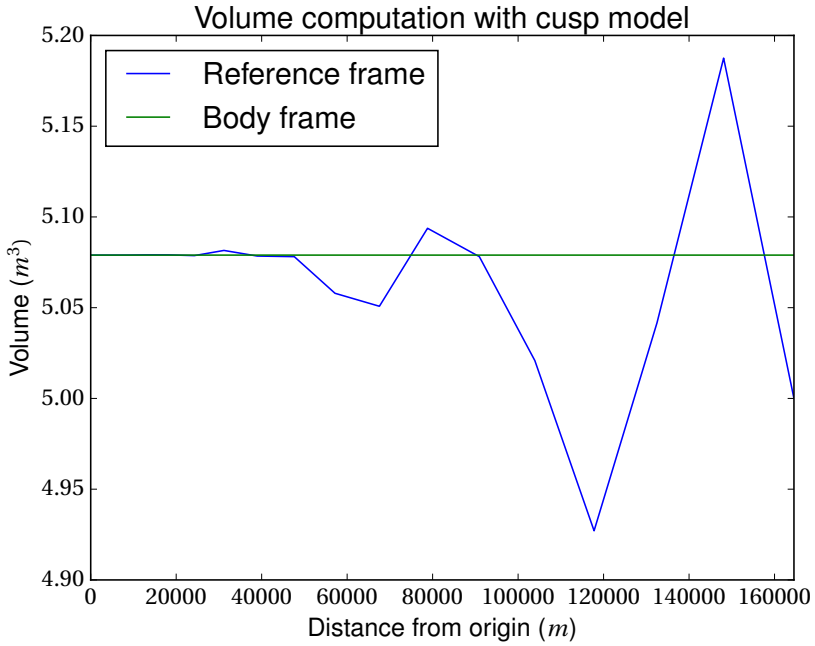


Figure 5.4: The volume for the same volumetric overlap as a function of its distance from origin. When the cusp model is naively used in the reference frame significant errors are introduced by the floating point arithmetic. Note that the ratio between volume and distance is the important factor here, but with floating point representations also the absolute values will determine the exact error, hence the absolute values are shown in the figure.

In implementations which use floating point arithmetic a significant problem remains with the cusp model. Since each simplex has one node at the origin, then when the geometries are far from the origin $|V_i| \gg V$ in eq. 5.10 and significant errors are introduced in the floating point arithmetic. For polyhedral DEM there is however a simple solution, namely that the intersection computation is done in the body frame of one of the particles. Figure 5.4 shows the volume computation as a function of the distance from origin. For the reference frame computation the error increase with the distance

whereas it remains constant for the body frame computation since in that case the computation is effectively always done in the left side of the figure i.e. close to the origin.

5.4 Multiple contact error

It is important to highlight that with the above presented methods multiple contacts are not properly resolved. Namely each particle-particle contact is always modelled as a single contact, which by the non-linearity of the force models, and also torque, is not correct. However, one should note that all other DEM methods does the opposite error, which is that it models single particle contacts as multiple contacts. However, since the force models are on such a simple form, the basic motivation why this is not properly resolved is that large modelling errors already exist. Moreover, for the typical use cases of DEM, merging the contacts should not be a large concern, but if for example more complex shapes such as toroidal particles is studied, this modelling error should potentially be further investigated.

5.5 Broad phase contact detection

Besides resolving the exact intersection between the particles, a DEM solver also needs a so called broad phase contact detection algorithm, as outlined in section 3. Where broad phase contact detection refers to effectively, but approximately, finding all the potential particle pairs. This is to avoid the quadratic scaling to the number of particles of the naive approach (which is to check all possible particle pairs). Typically accelerated data structures are used to improve the scaling. The two preeminent methods for this is the Cartesian grid and the bounding volume hierarchy (BVH). The Cartesian grid divides the space into cells and each particle is classified according to which cells it covers. Now all potential pairs can be resolved by only checking particles in the same or neighbouring cells. The BVH on the other hand is a tree data structure where each node is represented by a geometric object that contains all of its children [35], and in DEM the leafs are represented by bounding volumes of the particles. Now potential contacts for a particle is efficiently resolved by traversing the tree structure.

For the case of polyhedral DEM on the GPU it has been reported in the literature [7] how the grid implementation leads to severe performance degradation when the particle aspect ratio goes beyond 4:1. This degradation appears since the grid has to compromise between high memory usage and a more fine grained grid. Thus, when the aspects ratios are large the grid can be adjusted to roughly fit one of the smallest particles, but then the memory consumption can be significant. Thus to limit the memory consumption one has to increase the cell size, but then, as reported in the literature, the performance suffers since now more contact pairs are emitted from the data structure. The same problem also appears in cases where there is a few separated dense areas of particles, since then also the space between the dense areas needs to be covered by the grid. For the BVH this problem does not arise, here the memory usage is completely separated from the tightness of the bounding volume. Hence, with the

desire of a general framework, capable of handling varying environments and particle shapes, only the case of BVH broad phase contact detection will be considered in this thesis.

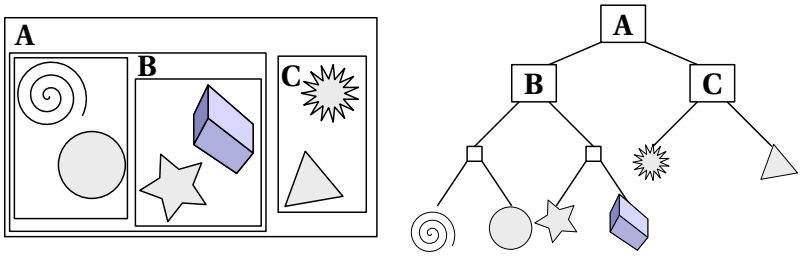


Figure 5.5: A Bounding Volume Hierarchy (BVH). Each node in the tree structure consist of a geometric object (here a rectangle) which covers all its children. This permits for fast overlap tests by traversing the trees structure from the root. Image by Schreiberx - Own work, CC BY-SA 3.0[36]

An example of a BVH can be seen in figure 5.5. The basic idea behind BVH is that the parent nodes of the tree can be tested with a geometric predicate, e.g. an overlap test, and if the predicate is true for a parent node one continues down to its children, however if a parent node does not fulfill the predicate all nodes beneath it can be disregarded. In this way, large portions of the tree structure can quickly be excluded from the search. For most realistic use cases one can typically obtain logarithmic scaling for each search query [35], albeit the worst case time-complexity is not improved from the naive approach. There are two main algorithmic aspects of a BVH structure, the build and the search algorithms, which both will be considered in the following sections.

5.5.1 Build algorithm

The build algorithm should given a set of geometric objects generate the tree structure of these objects and populate each node with its bounding volume. The build algorithm needs to be balanced according to; build speed, memory consumption (both temporary and permanent) and the quality of the final tree structure. While there is several different factors to consider for the quality of the tree structure, the short answer is that a higher tree quality yields faster search performance. To make the compromise between tree quality and build speed an important guideline is the build to search ratio. For instance, in scenarios with rigid geometries, the build can be precomputed once, whereas the tree can be traversed an indefinite number of times. Thus for rigid geometries, the speed of the build algorithm is of a rather low significance and mainly the quality of the tree structure should be of consideration. However, for contact detection of moving objects, e.g. particles in a DEM simulation as relevant for the current thesis, one typically only searches once for all objects of the tree against themselves. Thus, for the particles in DEM can the time spent on the build algorithm be directly weighted against the time saved in the traversal [35].

Mainly due to the advent of GPU-accelerated real-time ray-tracing, e.g. for computer games, there has been a significant advance in constructing BVHs on the GPU in the

past few years [37, 38, 39, 40]. To build the tree structure on the GPU often requires a significant different approach to the CPU-counterpart since the CPU algorithms rarely parallelize well. Most of the work has been performed for binary BVHs, i.e a degree of 2 per node. A notable early work is [37] where the feasibility of constructing the BVH on the GPU is shown. In this early work a simple build algorithm based on Morton codes was developed which was significant faster than CPU algorithms. This significant faster build could be achieved while still maintaining decent search performance. The work on binary BVH in [37] has been further extended in later articles optimizing both the quality of the tree and the build time [38, 39, 40]. It should be noted that some of these improvements are with regards to optimizing the Surface Area Heuristics (SAH), which in essence is minimizing the surface area of subtrees and are thus mostly relevant for ray-tracing applications.

A in-house FCC build algorithm developed by the author has been applied in this thesis work and the algorithm is based on some of the aforementioned binary BVH research [37, 38, 39, 40]. While the algorithmic details are not within the scope of this work, some aspects of the induced memory layout from the build algorithm will be of significance for later algorithmic considerations. Namely has the tree structure been adapted to higher degree nodes and especially has the memory layout been optimized for breadth first search. With the later meaning that each level of the tree is placed linearly in memory (as in [37]) with spatially close nodes also close in memory, both these efforts to improve the cache hit rate.

5.5.2 Search algorithm

The two most prominent algorithms for traversing the hierarchy is the breadth-first search (BFS) and depth-first search (DFS). BFS traverses the tree in a level-by-level manner, meaning that all active nodes are processed in a single level at once and only after is the next level considered. On the contrary, does DFS traverse a branch of the tree until termination, and at termination proceeds to the latest branch split. In the context of contact detection for real-time applications with serial implementations the most popular of the two have been reported to be the DFS. This preference for DFS is due to a larger amount of temporary memory is needed for BFS. Limiting this by some form of iterative implementation is not well suited for real-time applications, since consistent performance is often as important as the speed [35]. On the contrary, for DEM only the total simulation time is of importance, thus this limitation is of less significance.

The only relevant case for GPU accelerated search queries of a BVH is for the case where many queries are performed at once. Since otherwise at the root only a single thread will be active and the massive parallelism of the GPU is not properly utilized. Further, for the case of contact detection between objects (i.e not ray-tracing) it has been shown in the literature how a BFS implementation is superior in terms of efficiency [41].
In this work a BFS implementation is used. To constrain the memory the searches are made in batches which are iteratively adjusted to not breach the memory limitation. Due to the relatively slow changes in geometry in DEM this works well, and as previ-

Algorithm 2 Traverse a level, l, in a BVH and store the to be further processed nodes at next level $l+1$.

```
Let n = {} be the nodes at next level to further process.
for each node-search object pair, v, o, on level l in parallel do
  for each child, c, of the node v do
    if c intersects o then
      Let n = n ∪ {c}
    end if
  end for
end for
```

ously mentioned the uneven performance of this method is not a concern. A pseudo algorithm for each level traverse is seen in Algorithm 2. On a psuedo level this algorithm are very simple, however an efficient massively parallel implementation is a significant challenge. Especially packing the node, c, into the list, n, is significant to attain an efficient GPU BFS algorithm. This process is often referred to as stream compaction, which is for massively parallel implementations a non-trivial task [42], which incorporates to compute a prefix scan of the number of items to compact [43].

Also a DFS algorithm was implemented, but the batched BFS outperformed it by a factor of 5-10 times. By analyzing the algorithms with NVIDIA Nsight Compute[22] this large difference can largely be explained by a significant lower L1/2 cache hit rate of the DFS search.

5.6 World geometry contact detection

As outlined in section 3, it is desirable that the particles can interact with a surrounding geometry. In this work only the case of triangle mesh geometries are considered. As mentioned in section 4, the same contact forces for the particle interactions are used for the particle interaction with world geometry. This enforce some restrictions on the triangle meshes used for the world geometry, namely that their surface must be closed and have a defined interior.

Since the same contact forces are used as for the particle interactions, this also means that in large the contact detection remains the same. However, the number of triangles meshes are expected to be few, but to have many triangles, which is the opposite compared to the particles. Thus for the world geometries, each triangle is instead used in the same BVH implementation presented in section 5.5, this is in contrast with the particles, where a bounding volume of the entire particle were used in the BVH. Thus can the particle-triangle pairs be found by searching over the BVH containing the triangles. After computing the particle-triangle pairs, algorithm 1 can be applied and the same scheme for finding the properties as previously presented in section 5.3 is then followed. Note that for typical DEM applications the cost of finding, and resolving the particle-triangle pairs is small compared to particle-particle pairs, since the number of particle-triangle pairs will scale as bulk surface whereas the particle-particle pairs

scales as the bulk volume.

6

Physical Verification and Convergence of Method

This section presents the verification of both the method and the implementation. Since the force model is derived from contacts of spherical particles the model is first verified for two spherical particles colliding. Next, the convergence of this case with respect to particle resolution is studied. Then, the same experiments are repeated for an irregular rock shaped particle.

To further verify the method on a larger scale it is investigated whether relevant physical properties of particle systems can be captured. In particular we investigate whether the solver achieves convergence of the repose angle with respect to particle resolution, and in addition the solver is also verified against data from an experimental setup for rock material.

Due to DEM requiring substantial calibration experiments to properly validate it against experimental data, doing such a complete validation study against experiments are out of scope for this mainly method driven thesis. Thus only verification of the methods are considered.

6.1 Verification and convergence for spherical polyhedrons

Since the force models in use (see section 4) are derived from the spherical case the method is first verified for two spheres colliding. To average out the effect of discretization the experiments are repeated many times with the initial orientation of the spheres sampled according to a uniform distribution. Further, only the normal elastic force is considered. The radii of the spheres are $R = 0.1\,\mathrm{m}$ and the initial speed of both particles are $5\,\mathrm{m\,s^{-1}}$.

The spherical polyhedrons used for this experiment were generated by the built-in spherical triangulation tool in the software `IPS` - Industrial Path Solutions [44]. Figure 6.1 shows the generated polyhedrons for a few different resolutions. It should be noted that the models were normalized to give a consistent volume. This is not the case for the spheres directly after generation in the software `IPS` since the spheres are triangulated by placing the nodes on the surface of a sphere with a given radius, meaning that the volume decreases with the resolution of the triangulation from the models.

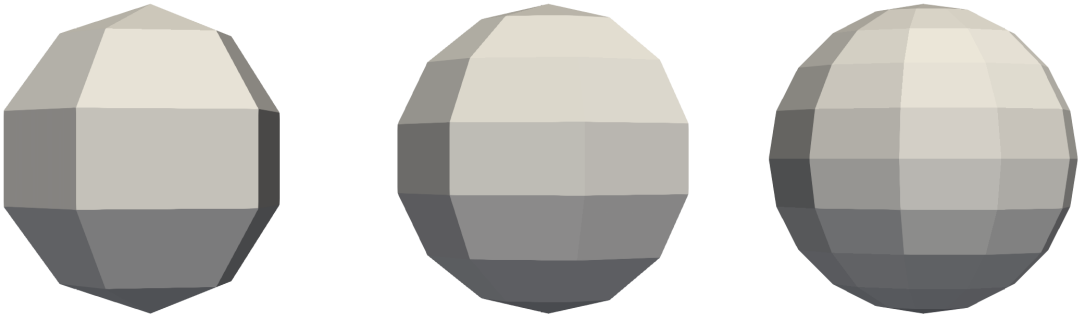


Figure 6.1: Spherical polyhedrons used for this study with the resolutions 40, 84 and 180 (left to right).

However, for the verification of the solver it is of a greater importance to keep the mass and volume of the particles consistent. This is to keep the initial kinetic energy consistent, since if not, it will skew the kinetic energy response studied in below experiments. In practical terms, a higher initial kinetic energy will result in a longer response.

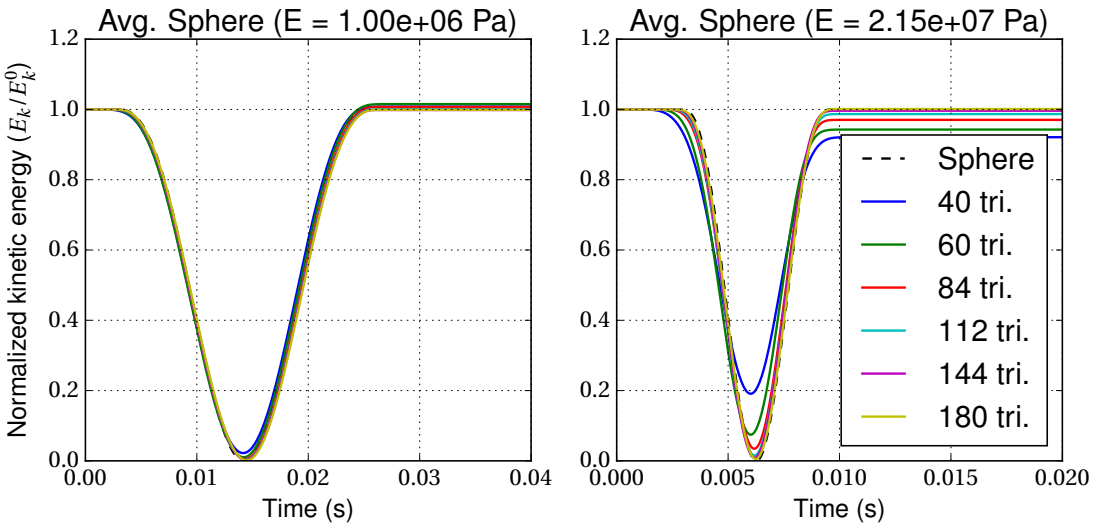


Figure 6.2: The normalized kinetic energy averaged over 10 000 instances as a function of time during collision of spheres. The dotted line shows the response for a perfect sphere, and the solid lines show the response for polyhedral spheres of different resolutions. The left figure shows the response for spheres with an Young's modulus of $E = 1\,\text{MPa}$ and the right figure for stiffer particles with $E = 21.5\,\text{MPa}$. The initial orientation of the polyhedrons are uniformly distributed.

Figure 6.2 shows the average kinetic energy as a function of time during a collision of two spheres. The kinetic energy, E_k, is normalized to the initial kinetic energy, E_k^0 (i.e. the kinetic energy of both particles before the collision) as E_k/E_k^0. In the figure to the left the Young's modulus is, $E = 1\,\text{MPa}$, and in the figure to the right the Young's modulus is $E = 21.5\,\text{MPa}$. In both cases the particle material has a Poisson's ratio of $\nu =$

0.25. The figures shows the response for different resolutions of the polyhedral spheres, and also for a perfect sphere simulated with the internal FCC spherical DEM solver. For the softer material properties it is clear that the response of all the polyhedrons closely match the perfect sphere with only slight deviations. However, for the stiffer particles the lower resolution polyhedrons leads to a significant error in the response. This is expected since stiffer particles will lead to less overlap and effectively making the overlapping volume resolution lower.

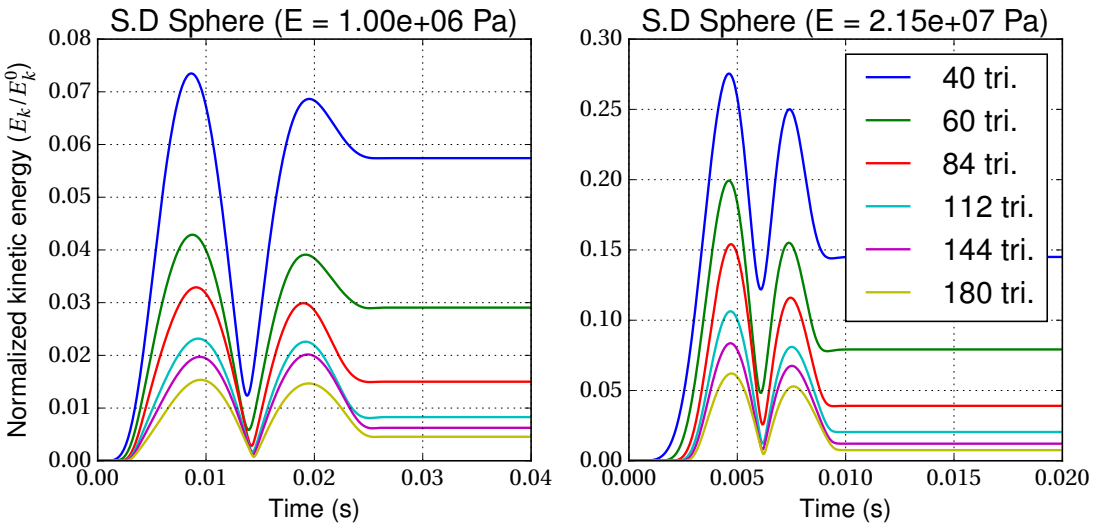


Figure 6.3: The standard deviation of the kinetic energy over 10 000 instances as a function of time for two spheres colliding. The left figure shows the response for spheres with an Young's modulus of $E = 1\,\text{MPa}$ and the right figure for stiffer particles with $E = 21.5\,\text{MPa}$. The initial orientation of the polyhedrons are uniformly distributed. Note that for a perfect sphere the standard deviation is always zero.

In figure 6.2 only the average response in the kinetic energy was considered, this to verify the force model and the method of using volumetric overlaps. However, as mentioned, by averaging out the response, the fluctuations induced by the discretization of the particles was neglected. Thus figure 6.3 also shows the response in the standard deviation of the normalized kinetic energy, which becomes a measure of the fluctuations induced from discretization. The two notable peaks in the figure show that the discretization cause fluctuations in the energy response, and that the fluctuations are most profound at the beginning and end of the collision. At the beginning and end of the collision, the overlapping volume is smaller, and hence will the effective resolution of the overlapping volume be lower, which cause larger fluctuations. Also, one can note that the standard deviation is higher for stiffer particles and that there is a clear reduction when the number of triangles are increased. However, there is a notable peak in the standard deviation for even the highest resolution polyhedrons, whereas for perfect spheres it should always be zero.

Finally, to get an overview of the convergence for a larger range of particle stiffness,

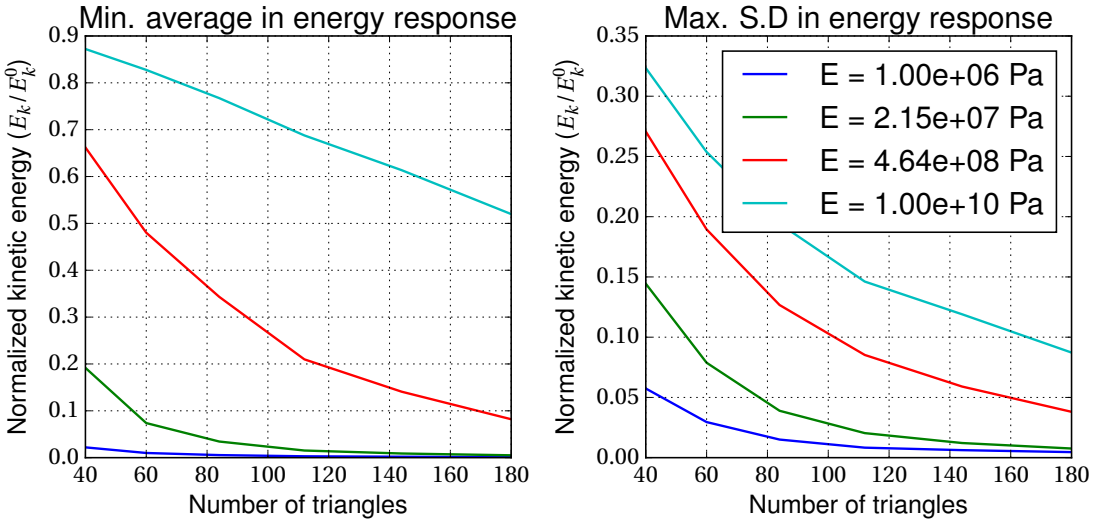


Figure 6.4: Convergence of the kinetic energy response for spheres with respect to the resolution of polyhedral spheres. To the left the minimum average normalized kinetic energy is shown and to the right the maximum standard deviation of the same quantity is shown. Clearly the convergence gets significantly slower as the stiffness of the particles increase.

the minimum value of the averaged normalized kinetic energy, i.e. the minimum of the data plotted in figure 6.2, and the maximum value of the standard deviation in the normalized energy response, i.e. the maximum of the data shown in figure 6.3, is considered in figure 6.4. Note that the maximum/minimum values are used instead of the average or RMSD over time, since in that case the measure become dependent of the length of the simulation, which has to be different to attain to the different stiffness. Hence to be able to compare between the different resolutions and stiffness, the extrema were considered. However, it should be noted, that these are not necessarily a precise measure of how well the response match the perfect sphere, but rather for comparisons between the polyhedrons. Nevertheless, figure 6.4 clearly show that there is a clear convergence in the response for the range of particle stiffness considered. Also it is clear from figure 6.4 that stiffer particles leads to larger discretization errors, not only affecting the fluctuations in the response, but also the average response.

This result is fundamental to polyhedral DEM. Foremost, that the method converge with respect to particle resolution, instead of an erratic response as is the case for the multispheres, is in itself profound. Also, a limitation of the method is here clearly shown, as the capabilities to model smooth surfaces with non-zero curvature is limited and in addition largely depends on the material properties. The results shows that the polyhedrons are to a larger extent capable of modelling smooth surfaces for softer particles than for hard particles.

6.2 Verification and convergence for irregular shaped particles

While the results for the spherical particles showed that the method in contrast to the multisphere method can converge in resolution, the use of spherical particles is however not the most relevant for actual usage of the method. In this section the method is studied for irregular shaped particles to ensure that the method can converge also for more complex shapes.

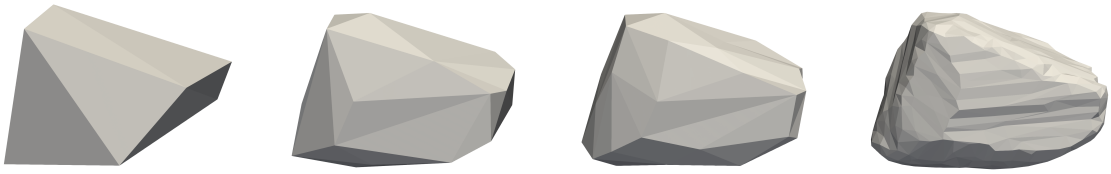


Figure 6.5: The irregular shaped polyhedral particle models used in this study. The figure presents particles with 10, 40, 70 and 1000 (original) triangles (left to right). While not clear from the figure several small concavities exist in the model.

Figure 6.5 shows the irregular shaped polyhedrons used in this study. The rightmost polyhedron is the original triangulation consisting of 1000 triangles. The three leftmost polyhedrons consisting of 10, 40 and respectively 70 triangles were generated by using the IPS - Industrial Path Solutions[44] triangulation simplification tool with the standard settings. The models were then normalized to give a consistent volume which the IPS simplification tool does not attempt to attain. Note that while not clear from the figure, the polyhedrons have several small concave sections.

The same numerical experiments as for the spheres were performed with the irregular shaped particles, namely studying the kinetic energy response during collision of two particles. Besides the changed particle model the experiments were conducted in the same way and with same parameter values as for the spherical particles.

The average normalized kinetic energy for the irregular shaped particles is shown in figure 6.6. The left figure shows for a Young's modulus of $E = 21.5\,\text{MPa}$ and on the the right $E = 464.0\,\text{MPa}$. The response for the irregular shaped particle is noticeable different opposed to spheres. For this particle model the minimum average kinetic energy does not approach zero, which is expected due to the irregular shape combined with the random initial orientation. Note that for an individual collision the kinetic energy will go to zero. The standard deviation of the normalized kinetic energy at different times are shown in figure 6.7. The standard deviation are significant higher compared to the spheres, which is expected since now the original shape will also induce fluctuations in the response. While there is a difference in the response for both the average and standard deviation when the resolution change, only small shifts can be seen and at least in the average response there is a tendency of convergence when the resolution is increased.

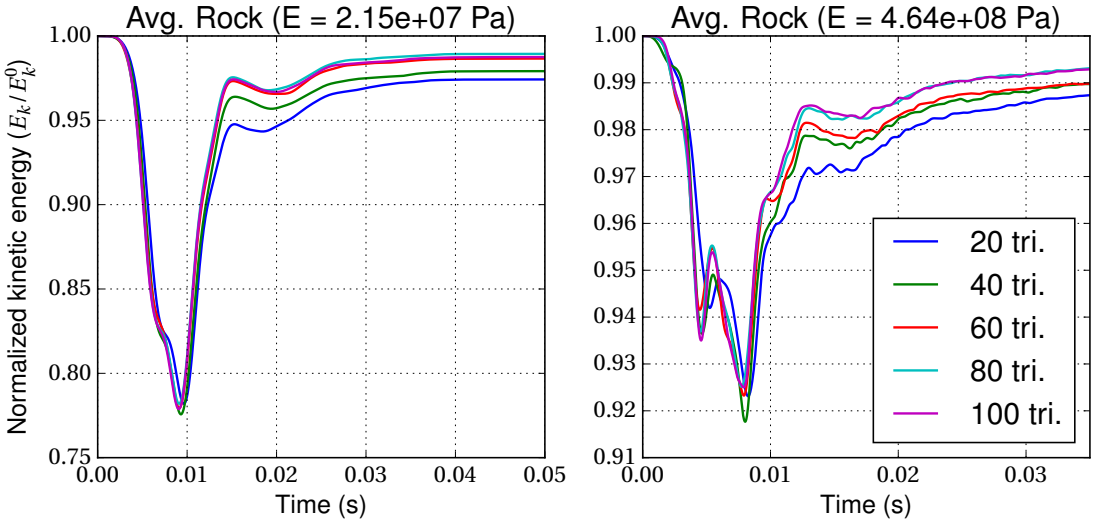


Figure 6.6: The normalized kinetic energy averaged over 10 000 instances as a function of time during collision of two irregular shaped particles. The different coloured solid lines show the response for different resolutions. The left figure shows the response for particles with an Young's modulus of $E = 21.5$ MPa and the right figure for stiffer particles with $E = 464.0$ MPa. The initial orientation of the polyhedrons are uniformly distributed.

Note that a similar analysis as for the spheres in figure 6.4 is not included. Since such a study could easily be misleading due to the more irregular shape of the response curves. Moreover, as will be further discussed in section 8, the actual convergence is not the important conclusion. The importance of these studies, rather is to highlight how utilizing volumetric overlaps of the polyhedrons leads to a force model that do not significantly change when small geometric changes are introduced, i.e. that it does not behave like multispheres or feature by feature polyhedral force models. This important property of the volumetric overlaps should already be evident from figure 6.6 and figure 6.7.

6.3 Verification and convergence on a laboratory scale

The numerical experiments in the previous sections to a large extent separated out solely the geometry of the particles and how changes to the geometry affected the the normal elastic response. Those experiments verified that polyhedrons with volumetric overlaps were well-behaved with respect to small geometric changes. However, to verify the tangential components of the force model and also study the aggregated effects of using polyhedrons, this section will study the method on a larger scale. The experiments are increased to a laboratory scale based on a calibration rig experiment. The experimental setup can be seen in figure 6.8, and both numerical and laboratory experiments were conducted as follows[45, 46]:

1. Particles were placed/generated in the hopper (the wedge shape at the top of

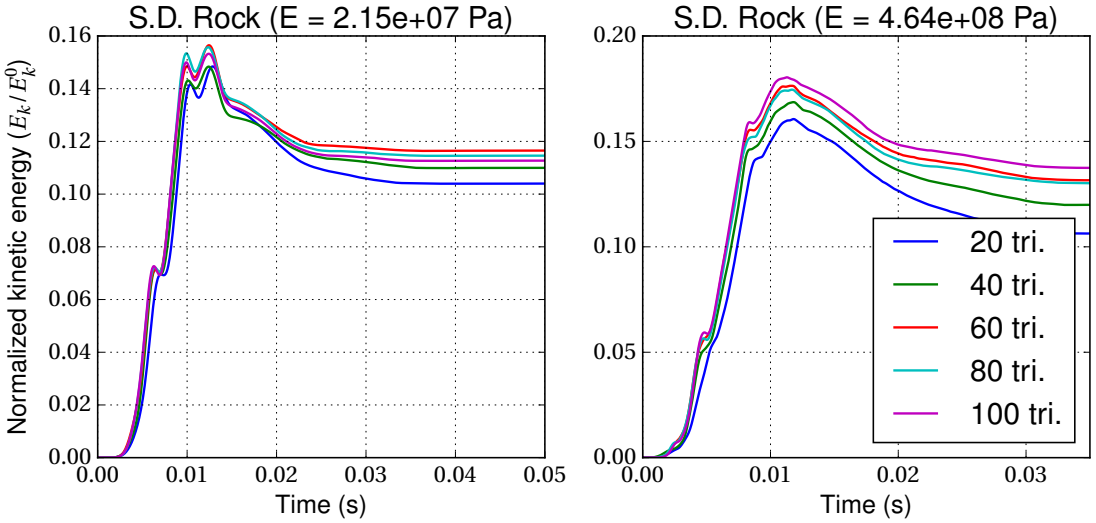


Figure 6.7: The standard deviation of the normalized kinetic energy from 10 000 instances as a function of time during collision of two irregular shaped particles. The response for different resolution particles are displayed as different coloured lines. The left figure shows the response for particles with an Young's modulus of $E = 21.5\,\text{MPa}$ and the right figure for stiffer particles with $E = 464.0\,\text{MPa}$. The initial orientation of the polyhedrons are uniformly distributed.

figure 6.8) with a given size distribution and specified total mass.
2. The trap doors below the hopper were opened.
3. Flow phase: Particles flow down from the hopper on to the tilted plane and finally to the bottom plate.
4. Steady state phase: Particles form a steady pile at the bottom plate. The repose angle is measured when the kinetic energy has converged to zero.

For the experiments, the repose angle was defined as the maximum angle of the pile. For each numerical experiment a 2-dimensional height map was created by dividing the particles into bins in the horizontal direction and measuring the maximum height in each bin. Then the repose angle was defined as the maximum derivative of a fitted sigmoid function to the height map of the system,

$$h(x) = \frac{c}{1 + \exp\{a(x-b)\}} \tag{6.1}$$

where a, b and c are constants fitted to each height map, $h_i = h(x_i)$, $i = 1,...,20$, by using the function `curve_fit` from the `python3` package `scipy.optimize`. Figure 6.9 shows an example of a height map with its corresponding sigmoid function.

In an earlier phase of this work, the numerical experiment on the calibration rig were conducted without any tangential spring stiffness and instead purely based on the force model proposed by Nassauer et. al. [9]. The simulation setup can be seen in table A.1. From such experiments it was clear that the desired behaviour of the system could not be achieved. Specifically, the particles did not reach a steady-state of non-zero repose angle. Instead small vibrations in the particles caused the repose angle to

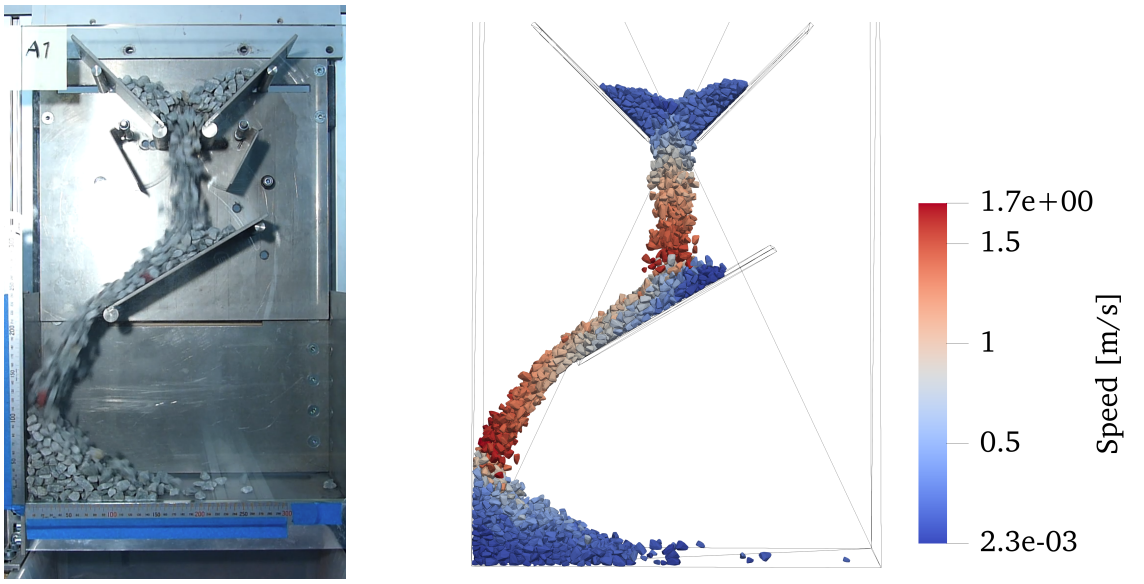


Figure 6.8: The calibration rig to measure the repose angle of rock material. To the left the real experimental setup is seen, to the right the rig modelled in the polyhedral DEM solver is shown. For the illustration of the DEM simulation the particles are colour coded according to their speed.

slowly drift towards zero. While there is a possibility of this being an issue with the parameter setup, a range of different friction parameters were investigated, but even if the simulation in other regards showed a highly unphysical behaviour due to high friction (e.g. the particles moving very slowly down the middle plate) there were still small vibrations in the pile of particles which lead to drifts in the repose angle. It should be mentioned that no studies were performed of this and no similar behaviours were reported in [9].

Since the desired behaviour of the system could not be achieved without the tangential spring stiffness a force model containing such component was derived from the force models for spheres (see section 4). Besides changed parameters from the use of different force models, the same setup as without a tangential spring was used for the simulation (see table A.2).

The response of the the calibration rig simulation with respect to particle resolution is seen in figure 6.10. Both the averaged kinetic energy during the flow phase of the experiment and the repose angle of the steady-state phase are shown. There are significant fluctuations in the repose angle even for a fixed number of triangles. This partly can be due to measurement errors from the fitted sigmoid function. However, all fitted sigmoid functions and their corresponding height map where manually inspected and there was a noticeable difference also in the height maps. Nevertheless, to verify that the variance measured in the repose angle was not entirely due to measurement errors, also the kinetic energy is shown in figure 6.10. The measurement errors in the kinetic energy should be minimal and yet significant fluctuations in the data can be seen. This indicates that fluctuations in the physics itself exist, which is expected since

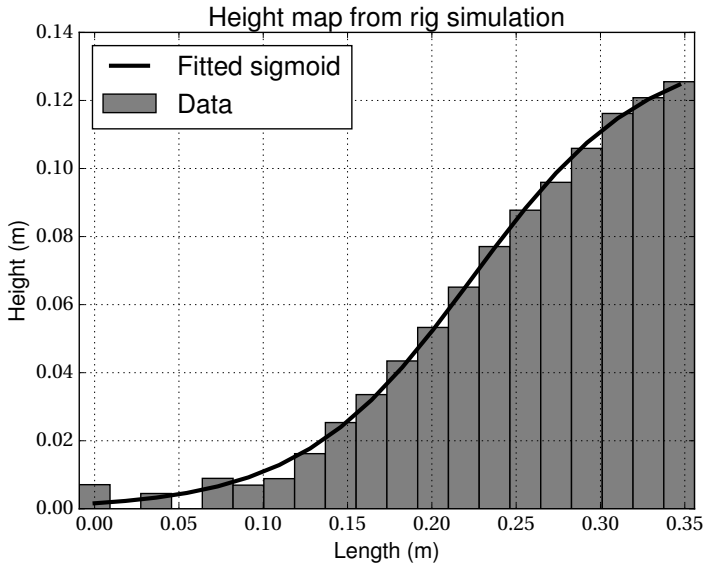


Figure 6.9: An example height map from the rig simulation. Also shown is a fitted sigmoid function to the height map. The repose angle is measured as the maximum derivative of the fitted sigmoid function.

the particles are generated randomly from a given size distribution. However, noticeable larger fluctuations can be seen in the repose angle compared to the kinetic energy. It is unclear whether this difference is due to larger measurement errors in the repose angle or induced from the physics, this uncertainty suggest that for future verification and validation of DEM better measurements than the repose angle may be found.

To compensate for these fluctuation a total of 11 simulations were performed per resolution. There is notable convergence for both the mean of the kinetic energy and repose angle. At around roughly 60 triangles per particle the mean of both measures roughly stabilizes within the range of the standard deviations. For the numerical experiments a stiffness of $E = 20\,\text{MPa}$ was used. One can note that the average response for the single collision for a similar stiffness shows a comparable convergence as for the calibration rig, at least in a broad perspective, where the single collision roughly have converged after 60 triangles, similar to the calibration rig.

Further, from figure 6.10 it is also clear that the most coarse grained particles at 10-20 triangles shows significant different values. This can intuitively be explained by the coarser particles having sharper edges. The sharper edges cause a higher degree of interlocking between the particles. The interlocking lowers the kinetic energy of the system during the flow phase due to lower flow rate from the hopper. Whereas for the repose angle the increase in interlocking makes the pile of particles more stable and causes the increase in repose angle. Further the repose angle can also be compared to experimental values, the experimental range of repose angles are between 33 to 42 degrees which are for a range of different materials and particle shapes. In figure 6.10 it is clear that only varying the particle resolution of one particle model almost covers the entire experimental range. Note that the experimental data was existing data which

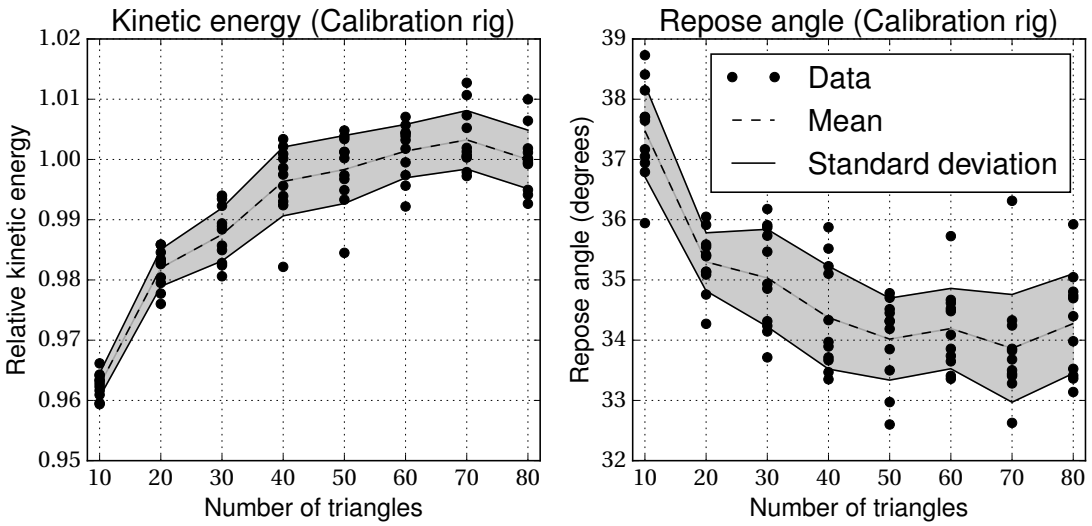


Figure 6.10: The kinetic energy during the flow phase of the calibration rig and the repose angle during the steady-state phase. Each dot shows a measure from an separate experiment. The dotted line shows the mean of the measures, and the grey area shows the range of the standard deviation. Noticeable fluctuations in both measures can be seen. This can be compared to the experimental range of 33 to 42 degrees in repose angle for a range of different materials and particle shapes.

were not sampled for this purpose and hence no stronger verification could be made. Rather the purpose of this verification was to consider if the general behaviour of the experimental setup could be captured by the method, which figure 6.10 clearly shows is the case. However, it should be highlighted that stronger verification and validation of the method should be one of the top priorities for further work.

Indirectly, figure 6.10 also demonstrates the need for DEM solvers to be highly efficient and capable of taking full advantage of modern hardware. Due to the high computational demands of DEM the open literature suggests that the typical studies are conducted on only a few data points, which figure 6.10 clearly shows can lead to significant misguidance. Hence to further enhance the usefulness and predictability of the method one of main research goals of DEM should be to advance the performance of the method. It is also evident that the software needs to have good scaling with regards to particle resolution since clearly the physics can be substantially altered if using too coarse grained models.

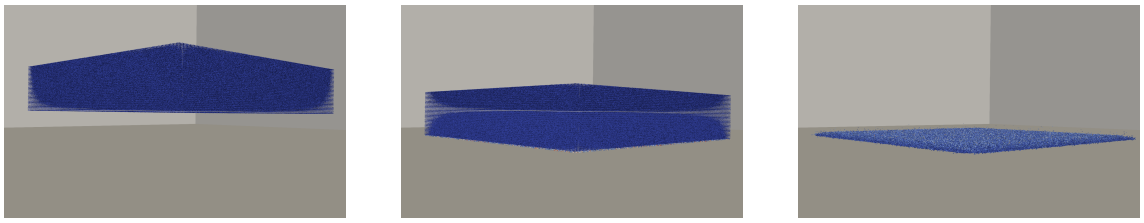
7
Performance

This section aims to answer several of the research questions related to the performance of the method and the implementation. Foremost, the general performance of the solver is analyzed and the key computational costs are identified. Further, the efficiency is compared to a spherical DEM GPU solver. Finally, the scaling of the solver with regards to particle resolution is characterized.

All performance measurements were conducted with a Nvidia Volta V100, and with double precision scalars throughout the implementation. The implementation also supports single precision computation, and gives roughly a 2 times speed up when using the V100 GPU.

7.1 Overall performance - gravity packing simulation

To investigate the overall performance, the polyhedral solver is compared to the spherical DEM GPU solver from FCC. For this the so called gravity packing case will be used. Gravity packing consist of letting an initial configuration of particles pack by gravity to a steady state configuration on a horizontal plane. This is an often applied benchmark case in the DEM community[7, 11, 25]. There are several different parameters which will significantly affect the performance. Henceforth to have any comparisons with regards to performance it is crucial that these parameters remains known and consistent. Table A.3 presents the simulations parameters used in this study chosen to mimic the case used in [11] to allow for a comparison. The same parameters are used for the spherical solver, with the notable exception of an additional rolling friction of $\mu_r = 0.5$



(a) $t = 0.0\,\mathrm{s}$ **(b)** $t = 0.4\,\mathrm{s}$ **(c)** $t = 1\,\mathrm{s}$

Figure 7.1: Illustration of the gravity packing simulation of 1×10^6 Schönhardt polyhedrons. The left figure shows the initial configuration, the middle figure shows the first particle impacts, and the right figure show when the last particles impact the particle bed.

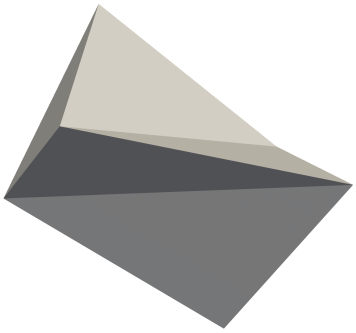


Figure 7.2: The Schönhardt polyhedron which is used for the gravity packing performance benchmark. The polyhedron is non-convex and consist of 8 triangles.

to avoid that the spherical particles rolls away and hence resulting in much fever pairs. An illustration of the gravity packing simulation can be seen in figure 7.1. In both simulations the number of particles is 10^6 and for the polyhedral solver the non-convex Schönhardt polyhedron is used, see figure 7.2, consisting of 8 triangles.

Figure 7.3 shows the execution time per step for the polyhedral and spherical simulations. For the polyhedral solver the time spent on the different steps of the method are displayed, and also the total time. For the spherical solver only the total simulation time is disclosed. It is evident from figure 7.3 that resolving the polyhedral intersections is the dominating factor for polyhedral DEM, even for a polyhedron only consisting of 8 triangles. A notable remark is that the BVH contact detection is shown to be highly efficient, where previously in the literature GPU BVH implementations have been disregarded due to being to slow. Moreover, the line marked as ”History” shows the time spent for maintaining the history required for the tangential spring stiffness. Maintaining this history is clearly not a significant computational burden for polyhedral DEM. Also the additional cost of having a tangential history in the contact forces are not a concern either, due to the contact forces being a very small fraction of the total simulation time. This is in contrast with previous claims in the literature where the tangential spring stiffness have been omitted with the motivation of a large computational burden by maintaining the history [9]. An interesting remark is that for polyhedral DEM with volumetric overlaps the time spent on the contact forces are independent of the resolution. This is is not the case for multispheres or feature-by-feature polyhedral DEM. Hence is adding more advanced force models, such as a tangential spring stiffness, less of a concern when using volumetric overlap.

The varying computational times of the simulations becomes clear when comparing to figure 7.4, which shows the number of broad phase contact detection pairs in the simulations. It is clear from comparison between these figures that for both the polyhedral and spherical solver the leading parameter of the simulation time is the number of pairs. For the polyhedral solver this should be the case considering that resolving the intersections are the dominating factor. For the first 40 000 steps no pairs are present

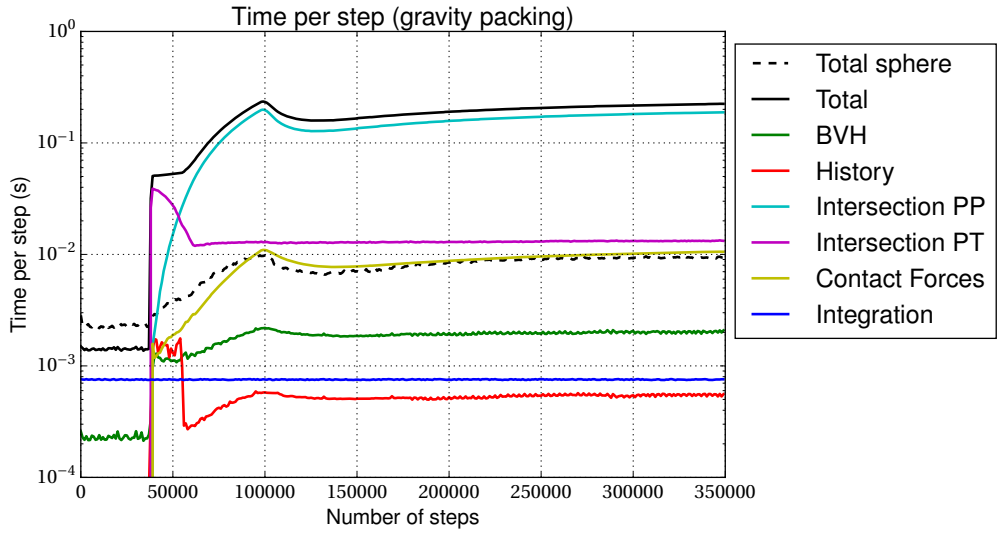


Figure 7.3: Execution time per step for gravity packing of 10^6 Schönhardt polyhedrons and spheres. The dotted line shows the execution time per step for the FCC spherical DEM solver. The solid lines shows the execution time for the different steps of the polyhedral solver, the solid black line shows the total simulation time. Figure 7.4 shows the number of collision pairs for the same simulation, note for both solvers how the execution time scales with the number of collision pairs. Both solvers used double precision scalars.

for either solver, here the polyhedral solver is even slightly faster than the spherical solver. This is due to a more effective broad phase contact detection, where the spherical solver uses a Cartesian grid instead of the BVH. However, at the end of the simulation when the number of pairs are high, the spherical solver is rather 20 times as fast as the polyhedral. However, the number of pairs of the polyhedral solver is significantly higher than for the spherical, the higher number of pairs are due to the changed particle model, which both change the coordinate number per particle and the overall physics of the systems. The difference in number pairs should not been largely altered by the different broad phase contact detection schemes, since the padding on the bounding boxes for the BVH was smaller than the padding used on each cell for the Cartesian grid. In [11, 13] the first second of a close to identical simulation with the Schönhardt polyhedron was reported to be an overnight simulation, this can be compared to our polyhedral solver where the first second takes roughly 1 hour with single precision and 2 hours with double precision, albeit with slightly faster hardware (roughly an expected 50% increase in our computational time with equivalent hardware).

Figure 7.5 shows the execution time per step for the gravity packing simulation against the number of collision pairs. Note that given a consistent number of pairs the polyhedral solver is roughly 10 times as slow as the spherical. There is a noticeable plateau of the polyhedral solver. The source of this is evident in figure 7.3 where the polyhedral-triangle mesh intersection is a large computational burden during the impact of the

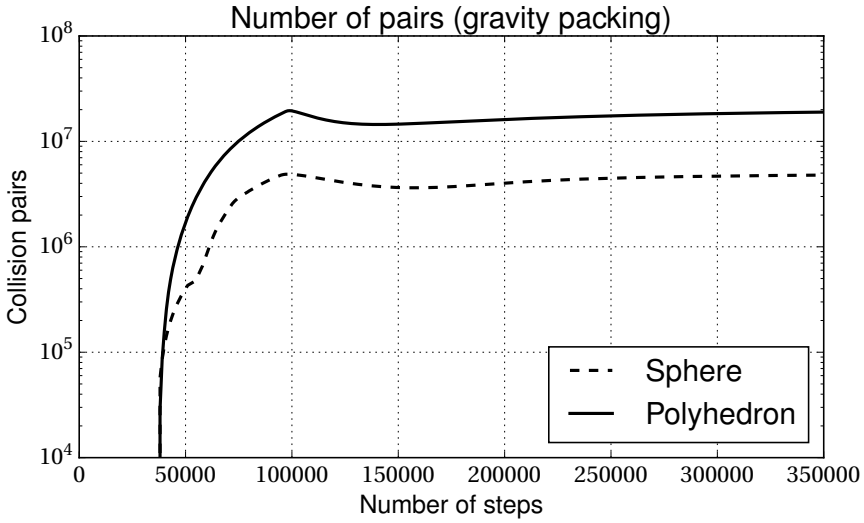


Figure 7.4: Number of collision pairs for gravity packing of 10^6 Schönhardt polyhedron and spheres. Can be compared with figure 7.3 where the simulation time for the solvers are displayed.

particles on the plane, but when a stable bed of particles have formed fewer particles reach the bottom plate. Hence mainly the scaling after the plateau is relevant for the polyhedral solver.

7.2 Scaling with particle resolution

In section 5.2 a simple heuristic algorithm was presented to minimize the triangles used in the expensive intersection routines. This section aims to investigate the implications of these efforts in terms of the scaling of the solver with regards to particle resolution. The basic heuristic claim is that due to the small overlaps present in DEM that good scaling can be achieved by such a simple filtering. First to give a fair comparison a synthetic test are performed where the impact of the changed particle models on the underlying physics are aimed to be minimized such that truly only the algorithmic performance are measured. However, also the calibration rig simulation described in section 6.2 is tested in order to asses the real implications of changed particle resolutions.

7.2.1 Scaling for single collisions

The simulation used as a synthetic test of the scaling is the same simulation as performed for the sphere verification and the initial investigations of irregular particles. Namely that two particles collide with no further collisions or interactions. However, to measure the performance accurately the GPU needs to be saturated. This is the case even for scaling benchmarks since the lower stress on memory can otherwise alter the result. Hence, many collisions are simultaneously performed to match the number

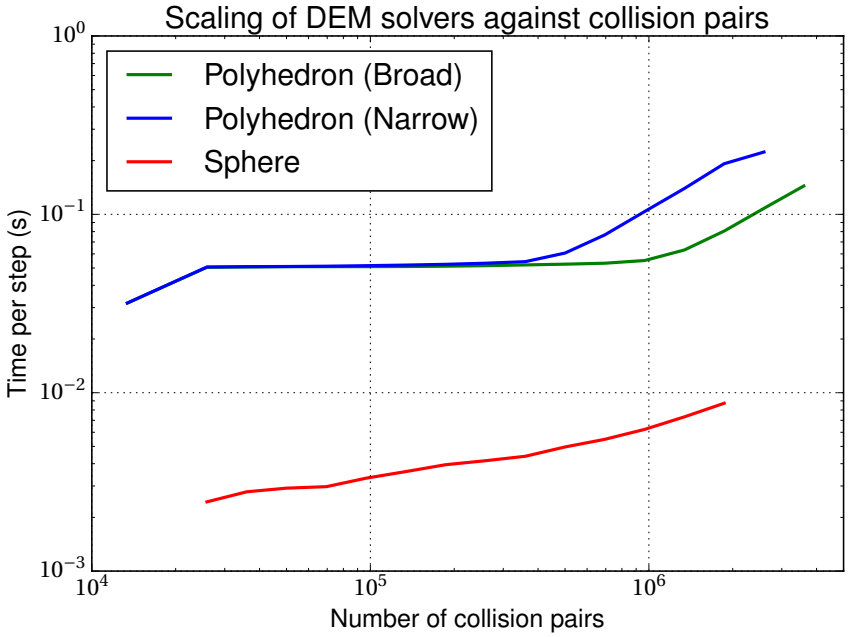


Figure 7.5: The scaling of the solvers against the number of contact detection pairs from the gravity packing simulation of 10^6 particles. For the polyhedral solver also the scaling against narrow phase contact pairs are shown, which are not relevant for the spherical solver.

of computational units of the used GPU. Further, only the irregular shaped particle is considered, this since the sphere was slow to converge for all but the softest particles, as can be seen in section 6.1. For example, figure 6.2 display how the lower resolution spheres give much shorter contacts, consequently would performance measurements on that case be severely skewed due to varying contact lengths. The irregular shaped particle on the other hand, quickly converged for all different values of the spring stiffness in its average response, and hence it is more suitable for the performance benchmark. Note that the lowest respectively highest stiffness used in this benchmark was not displayed in section 6.2, but the response for these material constants also had only slight deviations in the response for the different resolutions.

Figure 7.6 shows the performance of the irregular shaped particle for different resolutions. The left figure shows the relative performance, t/t_0, where t_0 is the execution time for the lowest resolution particle for respective algorithm. Both the filtering and actual polyhedral intersection is shown for different contact spring stiffness. The right figure shows the ratio between filtering of the triangles and the actual computation of the intersection properties. The relative cost of the filtering is low, where the cost is below 10% of the actual computation for almost all data points. This while the left figure clearly shows the effectiveness of the filtering, where for the stiffer particles close to linear scaling is achieved. It is also clear from the right figure that the intersection still has a quadratic component in the scaling, which also is predicted by the theory. However, the reduced constant factor in the quadratic scaling from the filtering is significant. It

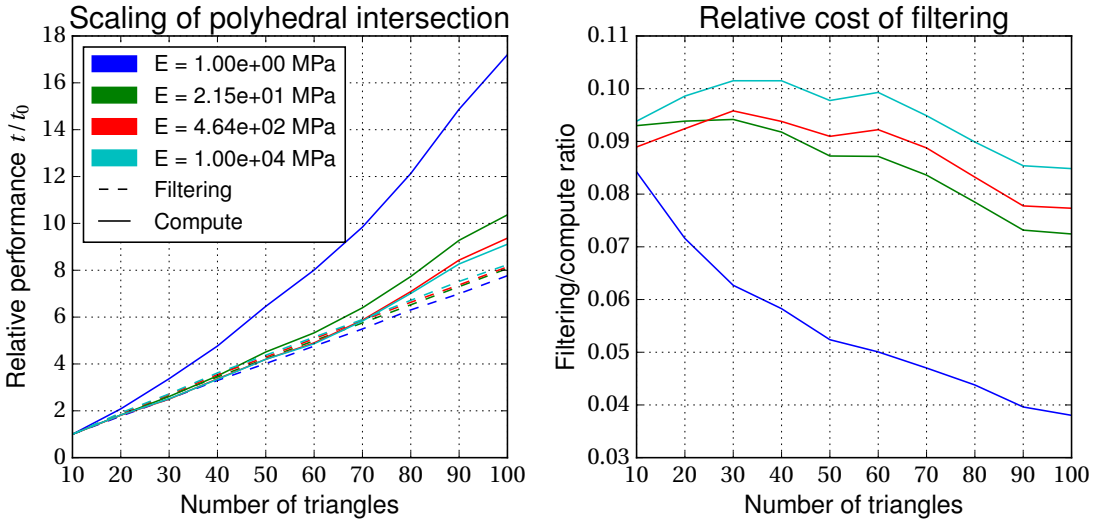


Figure 7.6: The performance of the polyhedral intersection for different resolutions of the irregular shaped particle. To the left is the relative performance against the lowest resolution particle for both the actual intersection computation and the filtering. To the right the relative cost of filtering against the actual computation is displayed.

is noteworthy that the lowest stiffness of 1 MPa is unrealistically low e.g. a spherical particle with the same mass and material properties give a maximal indentation depth of roughly one radius with the same initial kinetic energy.

7.2.2 Scaling of rig simulation

To further investigate the scaling of the solver with regards to particle resolution also the performance of the rig simulation was investigated. Figure 7.7 shows the scaling for the same simulations used to produce figure 6.10. Here the total simulation time is displayed and not only the execution time for the intersection computation. The scaling of the solver is close to linear with regards to particle resolution also for this more complex case.

7.3 Memory consumption

Besides being effective and having good scaling, a DEM solver also needs to effectively utilize memory such that adequate system sizes can be studied. This is especially true for GPU solvers, since in this case the memory is both more restricted and it cannot readily be increased since it is contained within the GPU itself. The primarily concern of this thesis was to attain an effective solver, but since memory allocation on the GPU is very ineffective, the performance optimization has also resulted in that the memory consumption of the solver is small. However, further decreasing the consumption is without doubt possible since few efforts were performed with regards to this.

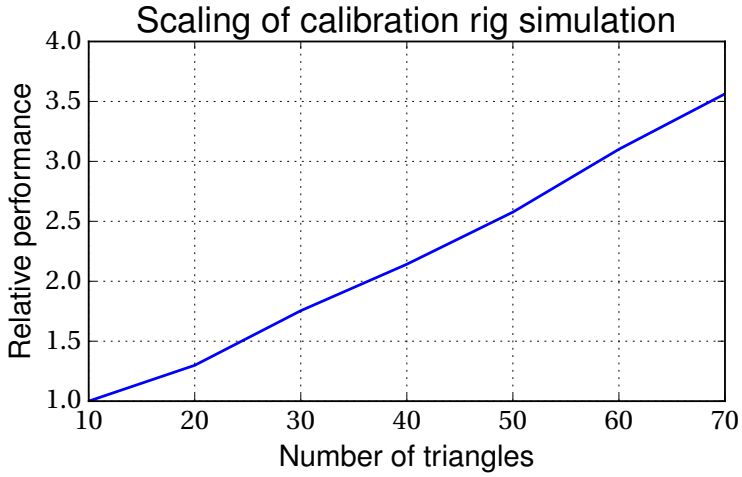


Figure 7.7: Scaling of the calibration rig simulation with the particle resolution.

Table 7.1 shows the memory usage for the gravity packing for a few different population sizes with the Schönhardt polyhedron. For a single NVIDIA Volta V100 GPU which has 32 GB of global memory, 10 million particles can be simulated with room to spare. However, the memory is dependent on the number of pairs and thus for sparse simulations a higher number of particles should be possible. For all simulations double precision scalars were used. Note that only a constant factor in the memory usage is dependent on the particle resolution, and hence will these values not be greatly altered by different resolution particles. Thus, simulating a billion of triangles on a single GPU should be within the scope of the presented implementation. For a similar physical fidelity, the highest polyhedral particle count in the open literature is 1 million particles on a single GPU [12], however this was conducted with a GPU only having 12 GB of global memory, and no reports where given of the actual memory consumption. In [7] a total of 10 million polyhedral particles were simulated on a single GPU with only 5 GB of memory, however in that work the physical fidelity was significant lower, which partly explain the low memory consumption. However, more importantly in [7] the efficiency of the solver was sacrificed to give lower memory consumption by always computing in parallel over the particles rather than over the particle pairs, which is not an effective solution on the GPU since this leads to significant divergence of the threads and, in addition, it results in each intersection being computed twice [25].

Table 7.1: Memory usage for gravity packing for a few different population sizes. Note that memory usage is specific to the GPU, i.e. the high memory usage for the lowest population sizes are due to allowing for constant factor allocations to scale with the size of the GPU. The consumption can be made smaller for GPUs with less memory/computational units. This is also the cause of sub-linear scaling of the memory consumption.

Population size:	1×10^5	5×10^5	1×10^6	5×10^6	1×10^7
Memory usage (GB):	1.1	2.2	4.0	11.7	20.6

8

Conclusion

The presented method and the implementation was shown in section 6 and 7 to be well-behaved and efficient, and this section aims to summarize and discuss the implications of these results.

It is clear from section 6 that the polyhedrons converge with respect to particle resolution, this by resolving the exact volumetric overlaps. Also clear from the difference seen in section 6.1 and section 6.2 the convergence rates are highly dependent of the shape of the original model. For the constant curvature model of spheres, the convergence was slow for realistic material properties, however for irregular shaped particles which have prolonged flat sections, the convergence was significantly faster. Moreover, similar convergence rate could be seen for the laboratory scale experiments in section 6.3 as for the single collision experiments of the same particle model in section 6.2. These convergence results are profound in several regards. Firstly, convergence for polyhedral methods utilizing volumetric overlaps have not previously been studied. Moreover, the main competing particle representation and currently most popular method, i.e. the multispheres, does not achieve this convergence.

It may seem like the convergence are mainly a topic of academic interest. This is to some extent true, since in e.g. simulation of rock material, only a few models are used to simulate all millions of particles and there are no evident arguments proposing that the original models is the best approximation of all the particles in the system. Rather, it is the implications of the convergence that are important, namely that small changes in the particle model only give small changes for the force model. This reduces the complexity of calibration by having a method that reacts to geometric particle model changes as one expects, i.e. it only changes the actual geometry and not contact stiffness. In turn, such a result has the potential of significantly reducing the time spent on calibration in DEM modelling. Further, having a contact force that is not affected by the geometric modelling can also enable further research on more advanced force models, whereas with multispheres any such efforts must also compensate for the erratic response from the geometric modelling, which will significantly hinder progress with the force models. Finally, the method enables the impact of particle shape to be properly studied. In contrast, for the multispheres it is not possible to determine if the changed particle system behaviour is a result of the particle geometric changes or a consequence of the changes in the contact forces the new model entails.

In section 7 it was evident that the presented method and the HPC GPU implementation is highly efficient. The polyhedral solver was for a consistent number of pairs only

roughly 10 times as slow as a state-of-the-art spherical solver. Moreover, the presented solver compared to the state-of-art polyhedral solver in the literature[13] shows a significant higher throughput, where the same case with their solver was an overnight simulation, whereas with our solver it could be simulated within an hour. From figure 7.3 is is evident that the number of contact pairs is the leading parameter for the execution time of the solver. This means that linear scaling with the number of particles are to be expected only for the case when the number of pairs scale linearly with the number of particles, which for large systems is typically the case. In section 7.2.1 and 7.2.2 it was shown that for relevant particle stiffness, near linear scaling could be achieved with regards to particle resolution. In addition, the filtering of triangles results in lower memory requirements which enabled higher resolutions than previously shown on the GPU in the open literature. Finally it was shown in section 7.3 that the implementation of the method is capable of simulating large system size, were for the gravity packing case 10 millions of particles could fit in a single GPU.

As previously mentioned, with a consistent number of pairs the polyhedral method with 8 triangles per particle is only 10 times as slow as a purely spherical approach. A reasonable conclusion is that there is roughly a 1:1 ratio for the computational time between number of triangles and number of spheres in a simulation, this due to the near linear scaling achieved with regards to triangles in the polyhedrons. The reason for this ratio not to be more favorable to spheres, is due to that each sphere-sphere pair needs to resolve the contact forces independently, whereas the contact forces only need to be resolved once per particle pair for polyhedrals. While this is a trivial conclusion for a purely spherical solver, this ratio will likely not be largely altered between a polyhedral and multisphere solve since each sphere-sphere pair also is resolved in the multisphere approach. This means that the increased geometric complexity of polyhedrons will likely be compensated for by having to resolve fewer contact forces in comparison with multispheres. With the more robust and well-behaved force model of polyhedrons, the potential advantage of the multispheres would be to have significant higher throughput, but tentatively these results show that this may not be the case. Moreover, since the computational cost of the contact forces scale with the resolution of the particles for multisphere DEM, its potential advantage over polyhedral DEM can be further diminished if more advanced force models are introduced.

8.1 Future work

Significant work remains in the field of DEM and, in particular, for polyhedral DEM. The most urgent work that remains for the presented method in this thesis is to further validate the method against experiments. Further, the GPU DEM framework developed in this thesis can easily be adopted to a multisphere approach, doing this can to a large extent determine the future path of DEM modelling, since as previously mentioned a continued usage of multisphere DEM, at least for most realistic particle shapes, could only be motivated if the performance of multispheres are significantly higher than polyhedrons.

The flexible representation and the well-behaved force models of polyhedrons will also

enable further work to attain more predictive DEM. For instance the difference seen when using different resolution particles in section 6.3 tentatively cast doubts into the single/few particle model approach typically used in DEM. To remedy this, an interesting future research path is adding noise to the particle models. With polyhedral particle representations this could be attainable with a single base model and then creating the noise by procedural methods (which is a popular method in CGI and computer games[47]). Such an approach would circumvent the memory usage issues of having many models. Further, a robust force model for irregular shaped particles should enable further research with regards to more advanced force models. The model used in this study to a large extent follows from the force model for smooth and regular shaped particles which was developed by Hertz in 1882 [28]. With help of modern technology, it is nor far fetched that better force models can be formulated, especially for e.g. the non-smooth behaviour of rock material. Also the integration schemes of the method should be further validated and compared to other approaches. Where possibly high-order explicit schemes could give benefits, but it would also be of interest to explore implicit methods.

Significant work also remains to attain even more efficient DEM implementations. Since the scope of this thesis was to implement a complete polyhedral solver, each individual step have not been fully explored, and without doubt, purely algorithmic improvements are possible for all stages of the method. Foremost, for polyhedral DEM, attaining more efficient intersection algorithms are of essence, where more research on this topic could potentially give significant gains in terms of performance. For DEM in general, the current approach of a single time step for the entire particle system is doubtful. In other particle methods multistep approaches have been explored and shown to give significant speed ups[48], this approach should also be explored for DEM. Also efforts to further take advantage of modern hardware should be explored. For GPU implementations, using the tensor cores which now takes up a significant area of the GPU should be one of the top priorities. Another approach is to utilize the flexible hardware of field programming gate arrays (FPGA) which is increasingly popular in HPC applications. For instance, such hardware has shown to outperform GPU implementations in molecular dynamics simulations[49], which is largely similar to DEM. This approach is attractive, since issues such as potentially unusable tensor cores on the hardware is avoided. However, with the current overnight compilation time, the engineering cost of FPGA programming is significant, but rapid progress is still being made in the underlying compiler technology [50] and FPGAs can thus quickly become more relevant.

Bibliography

[1] P. A. Cundall and O. D. L. Strack. “A discrete numerical model for granular assemblies”. In: *The Essence of Geotechnical Engineering: 60 years of Géotechnique*, pp. 305–329. URL: https://www.icevirtuallibrary.com/doi/abs/10.1680/ege.35362.0025.

[2] Rimantas Kaianauskas et al. “Discrete Element Method in Simulation of Granular Materials”. In: *IUTAM Symposium on Multiscale Problems in Multibody System Contacts*. Springer Netherlands, pp. 65–74. URL: https://doi.org/10.1007/978-1-4020-5981-0_7.

[3] J. F. Favier, M. H. Abbaspour-Fard, and M. Kremmer. “Modeling Nonspherical Particles Using Multisphere Discrete Elements”. In: *Journal of Engineering Mechanics* 127.10 (Oct. 2001), pp. 971–977. URL: https://doi.org/10.1061/(asce)0733-9399(2001)127:10(971).

[4] Mehrdad Pasha et al. “Effect of particle shape on flow in discrete element method simulation of a rotary batch seed coater”. In: *Powder Technology* 296 (Aug. 2016), pp. 29–36. URL: https://doi.org/10.1016/j.powtec.2015.10.055.

[5] H. Kruggel-Emden et al. “A study on the validity of the multi-sphere Discrete Element Method”. In: *Powder Technology* 188.2 (Dec. 2008), pp. 153–165. URL: https://doi.org/10.1016/j.powtec.2008.04.037.

[6] D. Höhner et al. “Comparison of the multi-sphere and polyhedral approach to simulate non-spherical particles within the discrete element method: Influence on temporal force evolution for multiple contacts”. In: *Powder Technology* 208.3 (Apr. 2011), pp. 643–656.

[7] Nicolin Govender et al. “Development of a convex polyhedral discrete element simulation framework for NVIDIA Kepler based GPUs”. In: *Journal of Computational and Applied Mathematics* 270 (Nov. 2014), pp. 386–400. URL: https://doi.org/10.1016/j.cam.2013.12.032.

[8] Benjamin Nassauer, Thomas Liedke, and Meinhard Kuna. “Polyhedral particles for the discrete element method”. In: *Granular Matter* 15.1 (Feb. 2013), pp. 85–93. URL: https://doi.org/10.1007/s10035-012-0381-9.

[9] Benjamin Nassauer and Meinhard Kuna. “Contact forces of polyhedral particles in discrete element method”. In: *Granular Matter* 15.3 (Apr. 2013), pp. 349–355. URL: https://doi.org/10.1007/s10035-013-0417-9.

[10] Nicolin Govender et al. “BlazeDEM3D-GPU A Large Scale DEM simulation code for GPUs”. In: *EPJ Web of Conferences* 140 (2017). Ed. by F. Radjai et al., p. 06025. URL: https://doi.org/10.1051/epjconf/201714006025.

[11] Nicolin Govender et al. “Hopper flow of irregularly shaped particles (non-convex polyhedra): GPU-based DEM simulation and experimental validation”. In: *Chem-*

ical Engineering Science 188 (Oct. 2018), pp. 34–51. URL: https://doi.org/10.1016/j.ces.2018.05.011.

[12] Nicolin Govender et al. "Large-scale GPU based DEM modeling of mixing using irregularly shaped particles". In: *Advanced Powder Technology* 29.10 (Oct. 2018), pp. 2476–2490.

[13] Nicolin Govender and Danile N. Wilke. "BlazeDEM-GPU for simulations where particle shape matters". In: *Proceedings of the 8th International Conference on Discrete Element Methods (DEM8)*. 2019.

[14] Nicolin Govender, Charley Wu, and Daniel Wilke. *Advances in computational mechanics using GPUs*. 2019. URL: https://developer.download.nvidia.com/video/gputechconf/gtc/2019/presentation/s9436-advances-in-computational-particle-mechanics-using-gpus.pdf.

[15] John Cheng, Max Grossman, and Ty McKercher. *Professional CUDA C Programming*. 1st. GBR: Wrox Press Ltd., 2014.

[16] NVIDIA Corporation. *Nvidia Tesla V100 GPU Architecture, The Worlds Most Advanced Data Center GPU*. 2017.

[17] NVIDIA Corporation. *Nvidia Turing GPU Architecture, Graphics Reinvented*. 2017.

[18] NVIDIA Corporation. *CUDA*. Version 10.1. URL: https://developer.nvidia.com/cuda-toolkit.

[19] Zhe Jia et al. "Dissecting the NVIDIA Volta GPU Architecture via Microbenchmarking". In: (2018). eprint: arXiv:1804.06826.

[20] Zhe Jia et al. "Dissecting the NVidia Turing T4 GPU via Microbenchmarking". In: *CoRR* abs/1903.07486 (2019). arXiv: 1903.07486. URL: http://arxiv.org/abs/1903.07486.

[21] Jeremy Appleyard. "CUDA Pro Tip: Optimize for Pointer Aliasing". In: (2014). URL: https://devblogs.nvidia.com/cuda-pro-tip-optimize-pointer-aliasing.

[22] NVIDIA Corporation. *NVIDIA Nsight Compute*. Version 2019.5.1. URL: https://developer.nvidia.com/nsight-compute.

[23] NVIDIA Corporation. *NVIDIA Nsight System*. Version 2020.2. URL: https://developer.nvidia.com/nsight-systems.

[24] Da Wang. "Accelerated granular matter simulation". In: 2015.

[25] J.Q. Gan, Z.Y. Zhou, and A.B. Yu. "A GPU-based DEM approach for modelling of particulate systems". In: *Powder Technology* 301 (Nov. 2016), pp. 1172–1182. URL: https://doi.org/10.1016/j.powtec.2016.07.072.

[26] Nicolin Govender, Daniel N. Wilke, and Schalk Kok. "Blaze-DEMGPU: Modular high performance DEM framework for the GPU architecture". In: *SoftwareX* 5 (2016), pp. 62–66. URL: https://doi.org/10.1016/j.softx.2016.04.004.

[27] H. Kruggel-Emden et al. "Selection of an appropriate time integration scheme for the discrete element method (DEM)". In: *Computers & Chemical Engineering* 32.10 (Oct. 2008), pp. 2263–2279. URL: https://doi.org/10.1016/j.compchemeng.2007.11.002.

[28] H. Hertz. "On the contact of elastic solids". In: *Z. Reine Angew. Mathematik* 92 (1881), pp. 156–171. URL: https://ci.nii.ac.jp/naid/10015562849/en/.

[29] R. D. Mindlin. "Compliance of elastic bodies in contact". In: *J. Appl. Mech.* 16 (1949), pp. 259–268. URL: https://ci.nii.ac.jp/naid/10003002284/en/.

[30] Eric W. Weisstein. “Polyhedron”. In: *MathWorld–A Wolfram Web Resource*. (). URL: https://mathworld.wolfram.com/Polyhedron.html.

[31] Bernard Chazelle. “An Optimal Algorithm for Intersecting Three-Dimensional Convex Polyhedra”. In: *SIAM Journal on Computing* 21.4 (Aug. 1992), pp. 671–696. URL: https://doi.org/10.1137/0221041.

[32] Timothy M. Chan. “A Simpler Linear-Time Algorithm for Intersecting Two Convex Polyhedra in Three Dimensions”. In: *Discrete & Computational Geometry* 56.4 (Apr. 2016), pp. 860–865. URL: https://doi.org/10.1007/s00454-016-9785-3.

[33] Rizwan Bulbul and Andrew Frank. “Intersection of Nonconvex Polygons Using the Alternate Hierarchical Decomposition”. In: July 2010, pp. 1–23.

[34] W. Randolph Franklin and Mohan S. Kankanhalli. “Volumes from overlaying 3-D triangulations in parallel”. In: *Advances in Spatial Databases*. Ed. by David Abel and Beng Chin Ooi. Berlin, Heidelberg: Springer Berlin Heidelberg, 1993, pp. 477–489.

[35] Christer Ericson. *Real-Time Collision Detection*. USA: CRC Press, Inc., 2004.

[36] Schreiberx - Own work. “Bounding Volume Hierarchy”. In: (2020). [Online; accessed May 05, 2020]. URL: https://commons.wikimedia.org/w/index.php?curid=17853864.

[37] Christian Lauterbach et al. “Fast BVH construction on gpus”. In: *Computer Graphics Forum* 28 (Apr. 2009), pp. 375–384.

[38] J. Pantaleoni and D. Luebke. “HLBVH: Hierarchical LBVH Construction for Real-time Ray Tracing of Dynamic Geometry”. In: *Proceedings of the Conference on High Performance Graphics*. HPG ’10. Saarbrucken, Germany: Eurographics Association, 2010, pp. 87–95. URL: http://dl.acm.org/citation.cfm?id=1921479.1921493.

[39] Kirill Garanzha, Jacopo Pantaleoni, and David McAllister. “Simpler and Faster HLBVH with Work Queues”. In: *Proceedings of the ACM SIGGRAPH Symposium on High Performance Graphics*. HPG ’11. Vancouver, British Columbia, Canada: ACM, 2011, pp. 59–64. URL: http://doi.acm.org/10.1145/2018323.2018333.

[40] Tero Karras. “Maximizing Parallelism in the Construction of BVHs, Octrees, and K-d Trees”. In: *Proceedings of the Fourth ACM SIGGRAPH / Eurographics Conference on High-Performance Graphics*. EGGH-HPG’12. Paris, France: Eurographics Association, 2012, pp. 33–37. URL: https://doi.org/10.2312/EGGH/HPG12/033-037.

[41] Sushil K. Prasad et al. “GPU-based Parallel R-tree Construction and Querying”. In: *Proceedings of the 2015 IEEE International Parallel and Distributed Processing Symposium Workshop*. IPDPSW ’15. Washington, DC, USA: IEEE Computer Society, 2015, pp. 618–627. URL: https://doi.org/10.1109/IPDPSW.2015.127.

[42] Markus Billeter, Ola Olsson, and Ulf Assarsson. “Efficient stream compaction on wide SIMD many-core architectures”. In: *Proceedings of the 1st ACM conference on High Performance Graphics - HPG ’09*. ACM Press, 2009. URL: https://doi.org/10.1145/1572769.1572795.

[43] Duane Merrill and Michael Garland. “Single-pass Parallel Prefix Scan with Decoupled Lookback”. In: 2016.

[44] Industrial Path Solutions. *IPS AB*. Version 3.9. Apr. 30, 2020. URL: https://industrialpathsolutions.se/.

[45] Johannes Quist and Magnus Evertsson. "Framework for DEM Model Calibration and Validation". In: Sept. 2015.

[46] Johannes Quist and Carl Magnus Evertsson. "Cone crusher modelling and simulation using DEM". In: *Minerals Engineering* 85 (Jan. 2016), pp. 92–105. URL: https://doi.org/10.1016/j.mineng.2015.11.004.

[47] Shaun Bangay. "Deterministic procedural generation of mesh detail through gradient tiling". In: *Proceedings of the Australasian Computer Science Week Multiconference on - ACSW '17*. ACM Press, 2017. URL: https://doi.org/10.1145/3014812.3014828.

[48] Yu Fang et al. "A Temporally Adaptive Material Point Method with Regional Time Stepping". In: *Computer Graphics Forum* 37.8 (Sept. 2018), pp. 195–204. URL: https://doi.org/10.1111/cgf.13524.

[49] Michael Schaffner and Luca Benini. "On the Feasibility of FPGA Acceleration of Molecular Dynamics Simulations". In: (2018). eprint: arXiv:1808.04201.

[50] Syed Waqar Nabi and Wim Vanderbauwhede. "FPGA design space exploration for scientific HPC applications using a fast and accurate cost model based on roofline analysis". In: *Journal of Parallel and Distributed Computing* 133 (Nov. 2019), pp. 407–419. URL: https://doi.org/10.1016/j.jpdc.2017.05.014.

A

Simulation parameters

Below the exact parameter setups of the calibration rig experiment and the gravity packing simulation is shown.

A.1 Calibration rig

Table A.1: Parameter values used for the calibration rig simulations without tangential spring.

Parameters:	Symbol	Values:	Unit:
Time step	δt	1×10^{-5}	s
Total mass of particles		1.0	kg
Particle size distribution (middle bounding box)		Normal ($\mu = 10$, $\sigma = 5$) truncated (low = 5, high = 10)	mm
Particle model		Irregular shaped particle (see figure 6.5)	
Young's modulus (P)	E	2.0×10^{7}	Pa
Young's modulus (W)	E	1.0×10^{8}	Pa
Possion's ratio (P)	ν	0.25	
Possion's ratio (W)	ν	0.25	
Friction static (PP)	μ_s	[0.1, 0.8]	
Friction static (PW)	μ_s	[0.1, 0.8]	
Friction kinetic (PP)	μ_k	[0.1, 0.5]	
Friction kinetic (PW)	μ_k	[0.1, 0.5]	
Transisiton velocity (PP)	v_t	[0.01, 1.0]	
Transition velocity (PW)	v_t	[0.01, 1.0]	

Table A.2: Parameter values used for the calibration rig simulations with tangential spring.

Parameters:	Symbol	Values:	Unit:
Time step	δt	1×10^{-5}	s
Total mass of particles		1.0	kg
Particle size distribution (middle bounding box)		Normal ($\mu = 10$, $\sigma = 5$) truncated (low = 5, high = 10)	mm
Particle model		Irregular shaped particle (see figure 6.5)	
Young's modulus (P)	E	2.0×10^{7}	Pa
Young's modulus (W)	E	1.0×10^{8}	Pa
Possion's ratio (P)	ν	0.25	
Possion's ratio (W)	ν	0.25	
Friction coefficient (PP)	μ	0.39	
Friction coefficient (PW)	μ	0.35	
Shear modulus (P)	G	2.0×10^{7}	Pa
Shear modulus (W)	G	2.0×10^{7}	Pa
Dampening coefficient (PP)	γ	0.453619	
Dampening coefficient (PW)	γ	0.382097	

A.2 Gravity packing

Table A.3: Parameter values used for the gravity packing simulations.

Parameters:	Symbol	Values:	Unit:
Time step	δt	1×10^{-5}	s
Number of particles		$\sim 1 \times 10^{6}$	
Grid size		$169 \times 169 \times 35$	
Time until first impact		0.4	s
Time until last impact		1.0	s
Particle model		Schönhardt polyhedron (see figure 7.2)	
Young's modulus (P)	E	2.0×10^{9}	Pa
Young's modulus (W)	E	1.0×10^{9}	Pa
Possion's ratio (P)	ν	0.25	
Possion's ratio (W)	ν	0.25	
Friction coefficient (PP)	μ	0.39	
Friction coefficient (PW)	μ	0.35	
Shear modulus (P)	G	1.0×10^{9}	Pa
Shear modulus (W)	G	1.0×10^{9}	Pa
Dampening coefficient (PP)	γ	0.453619	
Dampening coefficient (PW)	γ	0.382097	