Smooth and non-smooth approaches to simulation of granular matter

Stefan Hedman

Master’s thesis in Engineering Physics, 30 credits

*Supervisor:* Martin Servin, martin.servin@physics.umu.se  
*Examiner:* Krister Wiklund, krister.wiklund@physics.umu.se

Umeå University  
UMIT Research Lab  
SE-901 87 Umeå  
Sweden

October 17, 2011
Abstract

Granular matter is defined as a collection of particle grains, such as sand. This type of matter have different characteristics (solid, liquid and gas) depending on the energy level per grain. There are several approaches to modeling and numerical simulations of granular matter. They are used by different groups for different purposes, and the choice between the approaches is based on knowledge and tradition rather than what might be best for the purpose. The key questions are when to use what method and what physical quality is lost depending on the choice.

Two regimes of discrete element granular simulations emerge: smooth and non-smooth. To compare the efficiency and physical quality of the two approaches, four physics softwares are examined including Bullet Physics, LMGC90, AgX and LIGGGHTS. Test scenes are setup in each software and the results are compared to each other or to the results of other work.

The thesis is performed at UMIT Research Lab at Umeå University.

Keywords: granular material, contact dynamics
## Contents

1 Background ........................................ 2  

2 Purpose and goals .................................. 4  
  2.1 Limitations ........................................ 4  
  2.2 Disclaimer ......................................... 5  

3 Theory ............................................. 6  
  3.1 Notation .......................................... 6  
  3.2 Definitions ....................................... 6  
    3.2.1 Granular material .............................. 6  
    3.2.2 Smoothed-particle hydrodynamics .............. 7  
    3.2.3 Internal moment and stress tensor .......... 9  
    3.2.4 Strain rate ................................... 9  
    3.2.5 Inertial number ............................... 9  
    3.2.6 Sphere model and collision detection ....... 10  
  3.3 Smooth approach .................................. 10  
    3.3.1 Algorithm description ......................... 11  
    3.3.2 Linear dashpot force .......................... 12  
    3.3.3 Hertzian contact law .......................... 12  
    3.3.4 Tangential forces ............................. 13  
  3.4 Non-smooth approach ............................ 13  
    3.4.1 Regularization ................................. 15  
    3.4.2 Algorithm description ......................... 15  
  3.5 Efficiency ....................................... 16  

4 Method ........................................... 18  
  4.1 Simulation softwares .............................. 18  
    4.1.1 AgX Multiphysics (non-smooth) .............. 19  
    4.1.2 Bullet Physics (non-smooth) .................. 19  
    4.1.3 LMG90 (non-smooth) .......................... 19  
    4.1.4 LIGGGHTS (smooth) ........................... 19  
  4.2 Hardware ........................................ 19  
  4.3 Common parameters .............................. 19
4.4 Software-specific settings ........................................ 20
  4.4.1 Bullet Physics ........................................... 20
  4.4.2 AgX .................................................... 21
  4.4.3 LIGGGHTS .............................................. 21
  4.4.4 LMGC90 ................................................ 22
4.5 Testing framework ............................................. 22
  4.5.1 Bullet ..................................................... 23
  4.5.2 AgX ....................................................... 23
  4.5.3 LMGC90 ................................................. 23
  4.5.4 LIGGGHTS ............................................... 23
4.6 Quality estimation ............................................. 24
  4.6.1 Memory, $Q_M$ ......................................... 24
  4.6.2 Time, $Q_C$ ............................................ 25
4.7 Overview of test scenes ...................................... 25

5 Results ................................................................ 27
  5.1 TE1: Collision test ........................................... 27
  5.2 TE2: Shock propagation test ................................. 30
  5.3 TE3: Column compression test .............................. 33
  5.4 TE4: Pyramid stability test ................................. 36
  5.5 TE5: Box on an inclined plane ............................. 37
  5.6 TG1: Silo ..................................................... 39
  5.7 TG2: Cylinder dam break ................................... 41
  5.8 TG3: Rotating drum ......................................... 44
  5.9 TG4: Blade test ............................................. 48

6 Conclusions .......................................................... 51
  6.1 Quality features ............................................... 51
    6.1.1 Bullet Physics ......................................... 51
    6.1.2 AgX Multiphysics ..................................... 52
    6.1.3 LMGC90 ............................................... 52
    6.1.4 LIGGGHTS ............................................. 53
  6.2 Smooth or non-smooth? ..................................... 53
  6.3 Pipeline framework .......................................... 54

Bibliography .......................................................... 55
Chapter 1

Background

There are many ways of simulating granular materials numerically. The methods are used for different purposes in different groups. The engineering community often use discrete element methods (DEM), where each particle in the simulation is treated as a distinct individual. Engineers also use finite element methods (FEM), where the particles are considered as a continuous medium. For gasous granular simulations, Lattice-Boltzmann methods are often used.

Within the domain of DEM simulation, two types of approaches emerge; smooth and non-smooth [25]. The main difference in the approaches is how to handle contacts.

In the game and special effect simulations, the non-smooth DEM methods dominate. Here, the visual effect and computational speed is more important than the real physical behaviour. In the scientific studies of dense granular material both methods are used. There are hybrid approaches available; these provide non-smooth contact simulations but regularization parameters can make the materials slightly deformable like in the smooth simulations.

Granular simulations needs carefully designed methods to work. Grains far away from each other in a collection of particles may still be linked together by a complicated network of unilateral constraint forces. This is why the generating of force networks play an important role when comparing simulation methods. If the contact part of the algorithm is not accurate enough, force networks are either broken or too connected [11].

Granular materials are commercially important in applications as diverse as pharmaceutical industry, agriculture, and energy production. More than 50% of all products sold are either granular in form or involve granular materials in their production [12]. About 40% of the value added in chemical industry is linked to particle technology [2]. Despite this importance, the mechanics of granular materials is not well understood at present [6]. Research into granular materials is directly applicable and goes back at least
to Charles-Augustin de Coulomb, whose law of friction was originally stated for granular materials.
Chapter 2

Purpose and goals

The purpose of the project is to

- understand when and where non-smooth and smooth methods are beneficial for simulation of granular matter
- understand what the requirements from industrial applications mean for the choice of solvers and friction models in the non-smooth approach
- arouse curiosity in the non-smooth approach

The goals of the project are

- Overview and theoretical background to smooth and non-smooth approach to granular DEM simulations.
- Setup and run a number of test scenes in a number of smooth and non-smooth physics simulation software. Compare the results with respect to physical quality and computational efficiency.
- Analysis and conclusion. When and for what is smooth or non-smooth a suitable choice?

2.1 Limitations

The used particle shape is spherical and they have the same diameter in the tests (except the Blade test in Section 5.9). The spherical shape is chosen because not all softwares support all shapes. A non-spherical shape would be more close to reality - but we have to maintain compatibility with the softwares too.

Other limitations may be read in the rest of the report.
2.2 Disclaimer

It is easy to draw the conclusion that this is about benchmark testing of the softwares because of the memory and computation speed tests, but it is not. The purpose of the project is not to rank the softwares by performance, but rather to see for what purposes the softwares gives results of higher quality and for what cost.

When making the tests, choice of specific algorithms and other settings has been made (or use of the default in the respective software). These settings may not be the best according to the developer(s) of the software and in many of the cases guidance how to set these parameters were not available.

Since truly optimal parameter settings are out of reach, the results should not be used for strict comparisons to other results. The value in the results are the overall comparisons in between the softwares we are using (and to estimates of theoretical models), with the application on granular simulations.
Chapter 3

Theory

This chapter will introduce you to the notation used in the report, define important entities and compare the theoretical performance of the smooth and non-smooth methods.

3.1 Notation

Throughout the report, notation of vectors will be lowercase boldface letters. Matrices will be denoted using boldface letters and scalars with normal letters. If not stated otherwise, these vectors are column vectors. Position, velocity and acceleration will be denoted as \( \mathbf{x} \), \( \mathbf{v} \) and \( \mathbf{a} \), respectively. A single index of a vector will indicate that it belongs to the corresponding particle, i.e., \( \mathbf{x}_i \) refers to the vector position of particle \( i \).

Derivatives of variables are using dot notation. The \( n \)th derivative of a variable have \( n \) dots above it, e.g., \( \mathbf{a} = \dot{\mathbf{v}} = \ddot{\mathbf{x}} \).

We use the following notation for mean value and standard deviation. Sometimes an additional subscript can be added to emphasize that the mean is over that dimension, for example \( \langle E \rangle_t \) will be the mean of \( E(t) \) for \( t = t_1, t_2, \ldots \).

\[
\langle x \rangle = \frac{1}{N} \sum_{i=1}^{N} x_i 
\]

\[
s_x = \sqrt{\frac{1}{N-1} \sum_{i=1}^{N} (x_i - \langle x \rangle)^2} 
\]

3.2 Definitions

3.2.1 Granular material

Granular material is defined as a large collection of contacting solid, discrete, macroscopic particles. It is characterized by short-ranged and strongly dis-
Sipative forces in between the particles. The particles are large enough such that they are not subject to thermal motion fluctuations. The gravitational potential of a particle lifted its diameter is also much larger than its average kinetic energy. There is no upper size limit; the physics of granular material may thus be applied to asteroid belts or icebergs [8].

Examples of granular materials are sand and rice. Powders are a special class of granular material due to their small particle size, which makes them more cohesive and more easily suspended in a gas.

Granular materials does not have one single state of matter like many other materials have; it shows characteristics of solids (e.g. a sandcastle where grains are not moving), liquids (e.g. pouring of rice), or gases (dust, where grains have relatively large velocities). The kinetic energy level per grain can thus be a measure of the granular characteristics.

One system can show all states simultaneously in different regions, see Figure 3.1. The states of granular media can be characterized by the so called inertial number, defined in Equation 3.13

\[
I = \frac{m v^2}{k T}
\]

\[\text{Figure 3.1: Solid, liquid and gaseous states of granular media. The inertial number } I \text{ characterizes the different states. In the solid region } I < 10^{-3}, \text{ in the dense flow (liquid region) we have } 10^{-3} < I < 10^{-1} \text{ and on the flow surface (collisional flow) we got } I > 10^{-1}.\]

### 3.2.2 Smoothed-particle hydrodynamics

To calculate stress tensor, strain rate, inertial number etc. for arbitrary points in our granular media, SPH (smoothed-particle hydrodynamics) kernel approximation is used. Discrete point data \( f(x) \) and its gradient \( \nabla f(x) \)
is approximated using

\[ f(x) = \sum_{j \in \Omega} \frac{m_j}{\rho_j} f(x_j) W(|x - x_j|, h) \]  

(3.3)

and

\[ \frac{\partial f(x)}{\partial x_\alpha} = \sum_{j \in \Omega} \frac{m_j}{\rho_j} f(x_j) \nabla_{x_\alpha} W(x - x_j, h), \quad \alpha = 1, 2, 3 \]  

(3.4)

where \( W(r, h) \) and \( \nabla W(r, h) \) are kernel functions defined in equations 3.6 and 3.7, respectively. \( x_\alpha \) denotes the \( \alpha \)th element in \( x \) (\( x, y \) or \( z \)) and \( \Omega \) is the collection of neighboring particles to particle at \( x \). \( m_i \) is the mass of particle \( i \) and \( \rho_i \) is the corresponding mass density which must be calculated on beforehand using

\[ \rho(x_i) = \sum_{j \in \Omega_i} m_j W(|x_i - x_j|, h). \]  

(3.5)

The kernel functions are used to “smooth out” the field on each particle by averaging it over several neighboring ones. The choice of kernel function may thus have some affect on the results. The \textit{poly6} kernel is widely used in other SPH applications [19], defined by

\[ W(r, h) = \frac{315}{64\pi h^9} (h^2 - r^2)^3 \]  

(3.6)

\[ \nabla W(r, h) = \frac{945}{32\pi h^9} (h^2 - r^2)^2 r \]  

(3.7)

Figure 3.2: SPH kernel approximation in 2 dimensions. The smoothing length \( h \) determines which particles are neighboring other particles. In 2D we have a circular smoothing area, and in 3D a spherical smoothing volume.
3.2.3 Internal moment and stress tensor

For calculation of the stress tensor, we consider the tensorial internal moment $M^i$ of each particle $i$ defined in [5], see Equation 3.8. Here, $c$ is a contact between two grains $i$ and $j$. $f^c_{ij}$ and $r^c_{ij}$ are the contact force and distance vectors between the particles, respectively. $f^c_{\alpha}$ is the $\alpha$th component of the force $f$ exerted on particle $i$ at $c$. $r^c_{\beta}$ is the $\beta$th component in the position vector of the same contact.

$$M_{\alpha\beta} = \sum_{c_{\epsilon i}} f^c_{\alpha} r^c_{\beta}, \quad \alpha, \beta = 1, 2, 3. \quad (3.8)$$

The $3 \times 3$ stress tensor matrix $\sigma$ of a packing of volume $V$ is a matrix defined by

$$\sigma_{\alpha\beta} = \frac{1}{V} \sum_{i\in V} M^i_{\alpha\beta}, \quad \alpha, \beta = 1, 2, 3. \quad (3.9)$$

From this, the pressure can be calculated as

$$p = \frac{1}{3}(\sigma_{xx} + \sigma_{yy} + \sigma_{zz}). \quad (3.10)$$

3.2.4 Strain rate

The strain rate tensor field $\dot{\gamma}(x)$ for a velocity field $v^\alpha = v^\alpha(x^\beta)$ in a coordinate system $\alpha, \beta = 1, 2, 3$ is defined as

$$\dot{\gamma}_{\alpha\beta}(x) = \frac{1}{2} \left( \frac{\partial v^\alpha}{\partial x^\beta} + \frac{\partial v^\beta}{\partial x^\alpha} \right). \quad (3.11)$$

and using SPH kernel approximation from Equation 3.4 to calculate smoothed values for this field, we get

$$\dot{\gamma}_{\alpha\beta}(x) = \frac{1}{2} \sum_{j\in \Omega_i} \frac{m_j}{\rho_j} \left( v^\alpha_j \frac{\partial W_j}{\partial x^\beta} + v^\beta_j \frac{\partial W_j}{\partial x^\alpha} \right). \quad (3.12)$$

3.2.5 Inertial number

The strain state of a dense granular flow can be characterized through its inertial number [26]. The inertial number $I(x)$ is defined by

$$I(x) = \frac{\dot{\gamma}(x)d}{\sqrt{p(x)/\rho(x)}} \quad (3.13)$$

where

$$\dot{\gamma}(x) = \frac{1}{2} \sqrt{\dot{\gamma}^\alpha_{\beta}\dot{\gamma}^\alpha_{\beta}}. \quad (3.14)$$

This number can be seen as the ratio between typical time of deformation $\delta^{-1}$ and the typical time of confinement $\sigma_{zz}^{-1}$. 

Stefan Hedman, steffe@steffe.se

10 October 17, 2011
Generally three regimes are distinguished. The first is the quasistatic flow,

\[ I < 10^{-3}, \quad (3.15) \]

the second is the dense flow (liquid),

\[ 10^{-3} < I < 10^{-1}, \quad (3.16) \]

and the last is the collisional flow (gaseous).

\[ I > 10^{-1}. \quad (3.17) \]

3.2.6 Sphere model and collision detection

The simplest model of a granular particle is a sphere. Particle collisions can easily be detected by looking at the overlap \( \xi \), defined by

\[ \xi_{ij} = R_i + R_j - |x_i - x_j| > 0 \quad (3.18) \]

where \( \xi_{ij} \) is the overlap size, \( R_i \) the radius of particle \( i \), and \( x_i \) the position of particle \( i \).

The computational problem of detecting intersection between two objects in space is called collision detection. Collision detection is implemented in every physics engine in some way and is in many cases the most computationally heavy part of the whole simulation algorithm [15].

3.3 Smooth approach

In the smooth approach, the grains are assumed to be deformable to some extent. Contacts between grains are modelled using penalty forces modelled as springs or potential functions \( V(r_i) \). See Figure 3.3. These functions may depend on the contact overlap size \( \xi_{ij} \), relative velocity \( u \) and/or reaction forces. No overlap implies no penalty force.
Figure 3.3: In the smooth regime, potential (spring) models are used to simulate contacts.

Molecular Dynamics (MD) is similar to smooth DEM though it is used in a whole other area of science. In the MD case, each molecule is represented by a point or sphere, and interactions are modelled with continuous potential functions. The important similarity in this context is the twice differentiable equations of motion [24].

The use of springs (potential functions) implies the need of a simulation timestep smaller than the elastic response time for numerical stability. Smaller timesteps implies more computational effort. The timestep size needs to be even smaller to simulate more stiff material. In the limit of infinitely stiff springs, an infinitesimal timestep is needed. This limits the smooth approach to simulations of material with finite elasticity.

The notation “smooth” refers to the motion of particles. The particle trajectories are twice differentiable with respect to time and velocity jumps are not considered. Hence, the “resolution” of a smooth DEM simulation is higher than in the non-smooth.

The explicit time integration makes this method is highly parallelizable for a specific timestep, which implies faster computation. The requirement of a small time step limits these possibilities though, since more timesteps have to be computed for a given simulation length. The greatest bottleneck here may be the collision detection algorithm, since it is said to be the most time consuming component in 3D geometric reasoning applications [15].

3.3.1 Algorithm description

This is a short description of the algorithm used in smooth simulations [23].

1. **Predictor** The Taylor expansions of current data is used to compute the positions and velocities of particles at time $t + \Delta t$. 
2. **Collision detection** All colliding body pairs are detected. This step is the most heavy one in the simulation, it affects the overall computation time.

3. **Force computation** \( \mathbf{F} = -\Delta \mathbf{V}(\mathbf{x}_i) \) and \( \mathbf{a} = \mathbf{F}/m \)

4. **Corrector** Integration of the equations of motion \( \mathbf{x}_{n+1} = \mathbf{x}_n + \mathbf{v}_n \Delta t + \frac{1}{2} \mathbf{a}_n \Delta t^2 + \ldots \). Both data from the predictor step and the force computation steps are used.

5. **Step forward** in time: \( t = t + \Delta t \)

### 3.3.2 Linear dashpot force

The force in normal and tangential directions \( \mathbf{F} = \mathbf{F}^n + \mathbf{F}^t \) are calculated for each contact in a simulation.

For example, using the linear dashpot method,

\[
\mathbf{F}^n = Y \xi + \eta^n \frac{d \xi}{dt},
\]

where \( Y \) is the Young modulus and \( \eta^n \) is a damping coefficient in the normal direction. These constants contributes with an elastic and a dissipative force to the contact. The dissipative force causes a decrease of the relative normal velocity after the collision by a factor \( e \). This factor is called the coefficient of restitution.

### 3.3.3 Hertzian contact law

Within the smooth regime, explicit computation of the force is done using e.g. the Hertz contact model. The Hertzian contact law for spheres in normal direction is described by

\[
\mathbf{F}^n = \frac{2Y \sqrt{R_{\text{eff}}}}{3(1-\nu^2)} (\xi^{3/2} + e \sqrt{\xi} \frac{d \xi}{dt}),
\]

\[
\frac{1}{R_{\text{eff}}} = \frac{1}{R_i} + \frac{1}{R_j},
\]

where \( Y \) is the Young modulus, \( R_{\text{eff}} \) the effective radius of the colliding spheres (defined in Equation 3.21) and \( \nu \) the Poisson ratio. \( e \) is a dissipative constant called restitution. The first term \( \xi^{3/2} \) gives an elastic behaviour in the collision, while the term \( \sqrt{\xi} \frac{d \xi}{dt} \) contributes with damping.
3.3.4 Tangential forces

The model by Cundall and Strack [7] described in Equation 3.22 agrees with Coulomb’s friction law (Equation 3.23) and was successfully used in many simulations.

\[ F^t = \text{proj}_{\mu|F^n|}(F^t_{i-1} - h\dot{k'}u'_i) \]  
\[ |F^t| \leq \mu|F^n| \]

3.4 Non-smooth approach

In non-smooth approaches, grains are assumed to be perfectly rigid and the elastic contact response in between particles is neglected. Interactions are described by unilateral shock laws (impulses) and coulomb friction. These equations are written as non-differentiable relations involving velocity jumps, percussions and thresholds. Numerically this means that larger timesteps, compared to smooth DEM, can be used and also that indefinitely stiff materials can be simulated.

![Timestep Comparison](image)

Figure 3.4: In the non-smooth approach, a larger timestep can be used. However, if it is not carefully chosen, artefacts may occur. One artefact may be that grains penetrate each other more than they should and give a characteristic compression of the granular mass. The figures shows a drum simulation in AgX with a moderately sized timestep (a) and the same simulation with a timestep that is too big (b).

The contact forces are calculated implicitly, which includes solving for unilateral constraint forces. The effort of solving these contact problems is bigger than in the smooth approach, but since we can use a larger timestep size we do not have to solve them as many times.
For relatively small systems, computational cost is not a problem, and therefore the non-smooth approach may be a better choice in games and other interactive applications. For larger systems, the geometrical equations will take up a larger portion of memory (in contrast to the smooth approach where this is not needed at all).

The contact solving part is characteristic for the non-smooth approach. In one timestep, several bodies in the simulation may be in contact and may also collide. The solving procedure is about propagating impulses from one body to another, and then to the rest of the contacting bodies [14]. The solving step in this type of application is usually done iteratively. During one iteration, the impulse from one body may be propagated to its neighbor. During the next iteration, the original impulse may be propagated to the next body. This suggests that a lineup of \( N \) bodies in contact needs \( N \) iterations to propagate the force from one end to the other. This is exactly why the non-smooth approach needs to construct the geometric equations while the smooth don’t. The geometry equations in this case is a contact network and can be visualized as in Figure 3.5.

In Figure 3.6, we can see the artefacts that occur by using a too large timestep. Similar artefacts emerges when using too few solver iterations. In general, the limit of infinitely small timesteps and infinitely many iterations should give the desired results corresponding to the software physics settings.
3.4.1 Regularization

Regularization is a well-established method for handling with ill-posed linear system of equations. Instead of solving the original equation, $Ax = b$, a slightly perturbed system is solved, $Ax' + \epsilon = b$. In many cases the slightly perturbed solution, $x'$, is close to the solution to the original problem in the least square sense that $|Ax' - b|$ is small in some relevant measure.

The effect of regularization on constraint multibody systems is a slight elasticity in the contacts, which makes the regularized non-smooth simulation more similar to the smooth.

3.4.2 Algorithm description

Short algorithm description used in the non-smooth approach.

1. Collision detection All contacts are detected by looking at the overlaps (Equation 3.18).

2. Impulse computation Impulse collision is imminent only if two bodies approach each other (which is not the same as a static contact). The impulses from collisions are added to the particles.
3. Formulation of the geometry equation The contact network of the system is formulated.

4. Force computation Contact forces are calculated for the whole system. This can be done iteratively using e.g. a Gauss-Seidel solver. When the solver converged to a solution, the algorithm moves to the next step.

5. Integration of the equations of motion $t = t + \Delta t$

3.5 Efficiency

The computational effort is important when designing an algorithm. The effort is most often presented as a function of the size of the problem. By comparing the algorithms regarding the computational effort, one can determine where smooth or non-smooth algorithms are more efficient.

A comparison between Molecular Dynamics and Contact dynamics algorithms presented by Brendel, Unger and Wolf [3]. This section is based on the very same discussion.

Since we do not want our particles in non-smooth simulations to tunnel through each other during one timestep, the timestep size must be a fraction of $r/v$, where $v$ is the characteristic relative velocity and $r$ is the typical radius of curvature. See Figure 3.6 (a). Each timestep requires about $N^{2/D}$ ($D$ being the number of dimensions, 2 or 3) force iterations, each of which takes a number of computations of order $N$. The computational effort a the
non-smooth simulation with \( N \) particles is therefore
\[
T_{\text{non-smooth}} \sim N^{1+2/D}T_{\text{real}}v/r, \tag{3.24}
\]
where \( T_{\text{real}} \) is the real time span to be simulated [3].

In the smooth approach, interactions are modelled with springs of a characteristic stiffness \( K \). In spring simulations, the timestep needs to be smaller than the inverse spring frequency, \( \sqrt{m/K} \), where \( m \) is the characteristic mass of a particle. See Figure 3.6 (b). The effort is linearly dependant on the number of particles in the system though, which brings in a factor of \( N \).
\[
T_{\text{smooth}} \sim NT_{\text{real}}\sqrt{K/m} \tag{3.25}
\]
By using Equations 3.24 and 3.25, one can see what method is more efficient in what domain. Note that the shape and slope of this curve is strongly dependent of implementation and the state of the simulated system.

Figure 3.7: Domains where non-smooth DEM and smooth DEM are more efficient, separated by a power law \( N^{4/D} \). Note that the slope of this curve depends on implementation details (such as collision detection) and the state of the simulated system.

In Figure 3.7, the reader can see in what approach is more efficient depending on the characteristics of the simulation.
Chapter 4

Method

A number of tests were made to evaluate a number of existing physics simulation softwares. To make an evaluation study of these, they need to be tested under same geometrical and physical conditions. Therefore, a number of test scenes are carefully designed and reconstructed in each software.

In this chapter, chosen softwares are presented and the test scenes are briefly described.

4.1 Simulation softwares

To make comparisons between methods and simulation approaches, four softwares were selected. Their logotypes are shown in Figure 4.1.

(a) Bullet Physics  (b) AgX Multiphysics
(c) LMGC90         (d) LIGGGHTS

Figure 4.1: Chosen simulation softwares.
4.1.1 AgX Multiphysics (non-smooth)

AgX Multiphysics is a simulation software for professional and industrial applications [1]. It is developed to both meet the stability and performance requirements for virtual reality simulators and visual effects, as well as the requirements on realism and precision in engineering applications and virtual prototyping. AgX is not only developed to simulate contact dynamics, but also kinematics, constraints and motors.

4.1.2 Bullet Physics (non-smooth)

Bullet Physics is an open source physics engine featuring 3D collision detection, soft body dynamics, rigid body dynamics [4], and various constraints. It is used in games and visual effects in movies.

4.1.3 LMGC90 (non-smooth)

LMGC90 is an open platform developed for simulation of collections of rigid bodies in interaction [16]. It is used in the science of granular matter in the study of force networks and rheology. No known engineering tools have been based on LMGC90 or the same algorithms.

4.1.4 LIGGGHTS (smooth)

LIGGGHTS is a smooth open source Discrete Element Method (DEM) Particle Simulation Software. LIGGGHTS stands for LAMMPS Improved For General Granular And Granular Heat Transfer Simulations. It is used in the science of granular matter and in engineering applications.

4.2 Hardware

To make the simulations, a desktop with a quadcore Intel i7 processor and a memory of 8 Gb was used. During a prestudy, the amount of simulation time and memory were considered sufficient for the purpose and goal of this project.

4.3 Common parameters

The software interfaces are quite different, but the goal is to have the same physical settings for each software so they can be qualitatively compared in the same manner.

The parameters focused on are gravity, coefficient of restitution, friction coefficients, mass, particle size and particle shape. Young’s modulus is also
important, since it controls how rigid the bodies are (we aim for rigid, or at least nearly rigid body simulation). See the specifications in Table 4.1.

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Notation</th>
<th>Value</th>
<th>Dimension</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gravity</td>
<td>$g$</td>
<td>9.82</td>
<td>m/s$^2$</td>
</tr>
<tr>
<td>Young’s modulus</td>
<td>$Y$</td>
<td>$10^9$</td>
<td>N/m$^2$</td>
</tr>
<tr>
<td>Poisson ratio</td>
<td>$\nu$</td>
<td>0.45</td>
<td>1</td>
</tr>
<tr>
<td>Coefficient of restitution</td>
<td>$e$</td>
<td>0.1</td>
<td>1</td>
</tr>
<tr>
<td>Friction coefficient</td>
<td>$\mu$</td>
<td>0.3</td>
<td>1</td>
</tr>
<tr>
<td>Solver iterations (where applicable)</td>
<td>$n_{iter}$</td>
<td>10</td>
<td>1</td>
</tr>
<tr>
<td>Timestep (non-smooth)</td>
<td>$\Delta t_{non-smooth}$</td>
<td>$10^{-2}$</td>
<td>s</td>
</tr>
<tr>
<td>Timestep (smooth)</td>
<td>$\Delta t_{smooth}$</td>
<td>$10^{-5}$</td>
<td>s</td>
</tr>
<tr>
<td>Particle density</td>
<td>$\rho$</td>
<td>$10^3$</td>
<td>kg/m$^3$</td>
</tr>
<tr>
<td>Particle diameter</td>
<td>$d$</td>
<td>0.01</td>
<td>m</td>
</tr>
</tbody>
</table>

Table 4.1: Common parameters for the test simulations.

4.4 Software-specific settings

The simulations could not be made in the same manner in the different softwares, because of differences in implementation, interpretation of parameters etc.. The most important deviations and specific settings are listed here. If other settings are used, it is mentioned in connection with the text. The rest of the settings are set to default in each respective software.

4.4.1 Bullet Physics

*Bullet Physics* trunk version 2.77 was checked out via Subversion [28].

**Callbacks** Callbacks to constrain position of spheres along one or two dimensions were added in some elementary tests. This should however not affect the test results.

**Broadphase** The broadphase algorithm *btAxisSweep3* was used. This should not affect physical result but it will clearly affect computational time and memory usage. *btDbvtBroadphase* could also be used, it is said that its algorithm is more efficient but Bullets’ version of it is poorly optimized.

**Contact breaking threshold** When two objects separate a distance specified by *gContactBreakingThreshold*, Bullet assumes they are not in contact any more, and removes their contact data from being processed by the contact solver. The default value for this threshold is 0.02, which is not suitable for objects in our tests. To change this
value, the Bullet source code needed to be altered. The value was changed to 0.002.

**Solver** The solver `btSequentialImpulseConstraintSolver` (iterative projected Gauss-Seidel) was used to solve simulations in a `btDiscreteDynamicsWorld`. The number of iterations in the solver was set to the default of 10. Other settings were set to default (such as warmstarting and the randomized order of iteration).

**Dynamics** Moving objects (such as the blade and the lifted cylinder) was set as kinematic. Particles were dynamic and other objects were static.

**Deactivation** Deactivation of dynamic objects was turned off.

**Damping** Damping (both linear and angular) was set to 0.

### 4.4.2 AgX

**Version** The version 1.10.2.0 was used and compiled with the current default settings, except that it was compiled with Lua support.

**Callbacks** Callbacks to keep the position of spheres along a line/in a plane were added (for some of the elementary tests). This should however not affect the test results.

**Solver** An iterative solver of projected Gauss-Seidel type was used and the number of iterations was set to 10.

**Threads** *AgX* was run with 4 threads enabled.

### 4.4.3 LIGGGHTS

**Version** The version used was “LIGGGHTS 1.2.7 based on lammps-10Mar10”.

**Particle interaction** Single distance interactions was used. This algorithm is said to be efficient in most cases.

**Pairwise neighbour list style** “Bin” style was used with bin size of 10% of the particle radius. In some tests, the bin size had to be changed to make *LIGGGHTS* perform better. In the rest of the report, this entity will be referred to as “bin size”.

**Pair style** A linear Hookian spring model was used.

**Timestep** A timestep of $10^{-5}$ is used if not specified in the test results.

**Integrator** The NVE integrator was used. NVE is also called the NVE ensemble and it describes a system with a fixed number of particles ($N$), a fixed volume ($V$), and a fixed energy ($E$).
Processor One virtual processor was used.

Force on mesh Forces on meshes (as in the Blade test in Section 5.9) were measured using the `fix mesh/gran/stressanalysis` command. The extraction of force was made once per millisecond, averaging over all time steps in between.

Particle composition To produce a body made of composite particle grains, the `rigid` command was used.

4.4.4 LMGC90

- Kinematic and static objects Driven degrees of freedom were added to make e.g. blade and moving cylinder kinematic. Static objects was also used this way, but the driven degrees of freedom were set to zero.

- Planes Since no boxes seem to exist in LMGC90, finite planes were added with faces directed to where they needed. In the blade test, two planes in opposite directions simulated the blade.

- Solver A non-linear Gauss-Seidel solver with 10 iterations was used. To push LMGC90 to use 10 iterations, a tolerance of 1000.0, minimum number of iterations 10 and convergence check frequency 10 was used. Relaxation was set to 1.0.

4.5 Testing framework

The test scenes were set up in an own data format. By creating wrapper classes that could run each software, the tests were easily run with same configurations. The programming language used was Python [22].

![Figure 4.2: The main pipeline class hierarchy. The four software-specific classes Agx, Bullet, Ligghits and Lmgc90 inherits properties and methods from the Software base class. The State class contains information about a simulation at a specific time, and can be saved to or loaded from file. The Dump class contains several States and have methods for higher-level data manipulation than a single State.](image-url)
The testing framework was constructed using object oriented code and the class hierarchy can be studied in Figure 4.2. Documentation and source code may be downloaded from [27].

The idea on how to use the pipeline framework is described in Algorithm 1.

**Algorithm 1** The data production pipeline.

```plaintext
1. for all test scenes do
2.   Read geometrical scene data
3.   for all simulation softwares do
4.     Run the scene in current software
5.     Store output data
6.   end for
7. end for
8. Post-process
```

The specific handling of the different softwares follow.

### 4.5.1 Bullet

Since Bullet runs in C++, a complete C++ application that could be started from a set of instructions was created. The Python application could create a temporary folder for input and output from the Bullet app. After the simulation, data could be read by Python and the temporary files were afterwards deleted.

### 4.5.2 AgX

AgX was controlled in mostly the same way as Bullet Physics. The difference was that AgX already had Lua scripting support available which made things easier. The Python app simply edited a Lua scene template file and ran them in AgX.

### 4.5.3 LMGC90

The Python bindings for LMGC90 was used to control the software. A template Python scene file was edited and executed, and the output files were parsed.

### 4.5.4 LIGGGHTS

LIGGGHTS uses an own type of script files, so simulation scripts needed to be created by the Python app. Some parts were read from a template file.
4.6 Quality estimation

To study how each physics simulation software performs with respect to computational time and physical behaviour, the following entities are estimated and presented.

- $Q_K$: kinematic (or rheology) measure
- $Q_F$: force distribution measure
- $Q_E$: energy measure
- $Q_C$: computational time
- $Q_M$: memory usage

We refer to these quantities as ‘quality measures’ although it quality is quite different for different applications.

The quality measures $Q_K, Q_F, Q_E$ are computed relative to a reference from experiment, analytical solution, benchmark simulation or other models - specific for each test scene. If no reference exist the smooth DEM solution is used for reference. Descriptions of how these entities are calculated is briefly described in Section 4.7 and fully describled before each result in the Results section.

$Q_C$ and $Q_M$ are calculated as described in the following sections.

4.6.1 Memory, $Q_M$

The memory usage recorded is the total memory used by each main application (also called virtual memory). In the results, the memory usage relative to the “empty simulation memory” is presented.

$$Q_M = M - M_0$$ (4.1)

The “empty simulation” is rather not empty; it contains one particle in empty space without gravity. This is added to make sure the software started simulation, collision detection and all other needed things to run. The memory usage by the softwares are listed in Table 4.2.

Note that the value of $Q_M$ can be negative if the software uses less memory than the “empty simulation”. This is not very probable though.

<table>
<thead>
<tr>
<th>Software</th>
<th>$AgX$</th>
<th>Bullet Physics</th>
<th>LIGGGHTS</th>
<th>LMGC90</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory</td>
<td>262</td>
<td>94.4</td>
<td>39.1</td>
<td>92.2</td>
</tr>
</tbody>
</table>

Table 4.2: Empty simulation memory usage (Mb) for each software.
4.6.2 Time, $Q_C$

The simulation time quality $Q_C$ is given as a fraction between the needed computation time $t_{comp}$ subtracted by the needed time for the “zero-test” $t_0$ divided by the real-world simulation time $t_{sim}$. The value for $t_{comp}$ is obtained by measuring the time taken by a software to run a simulation without visualization or extras on.

$$Q_C = \frac{t_{comp} - t_0}{t_{sim}}$$  \hspace{1cm} (4.2)

For example, a real-time simulation will have $Q_C \leq 1$. A simulation slower than realtime will have $Q_C > 1$. The term $t_0$ is expected to be negligibly small compared to $t_{comp}$.

4.7 Overview of test scenes

Below is a short description of each test scene. A full description can be found in Chapter 5.

The tests are labelled with three characters. The first is T as in “test”, second is E for “elementary”, G for “granular” or M for “multidomain”. The third character is simply the test number within the category.

The test scenes were chosen to bring forth the three granular media states discussed in Section 3.2.1. They are also chosen so they can be constructed in all softwares.

TE1: Collision test  Particles are collided. Trajectories and energies are studied. The value of $Q_K$ is calculated by comparing trajectories. $Q_F$ is a binary quality measure, telling whether the collision went well or not.

TE2: Shock propagation test  Particles are lined up against each other, and another particle gives them an impulse from one end. Propagation of force is studied and $Q_F$ is set to a value corresponding to the propagation time. $Q_E$ is calculated by comparing to other simulation.

TE3: Column compression test  Particles are stacked and the behaviour is studied. Force quality $Q_K$ is calculated by looking at the average kinetic energy in the system and $Q_F$ is proportional to the total column height.

TE4: Pyramid stability test  Particles are stacked in a pyramid and the required timestep to make it stable is measured. $Q_K$ is a binary value; 1 if a stable pyramid could be produced and 0 if not.

TE5: Box on an inclined plane  The angle where the box is starting to slide is measured and compared to theory. $Q_K$, $Q_F$, $Q_E$ are omitted and the slide angle is presented instead.
**TG1: Silo** A silo is filled with particles. Vertical stress distribution is studied. $Q_F$ is calculated by comparing the measured stress distribution to a theoretical model.

**TG2: Cylinder dam break** A cylinder is filled with particles and then it is lifted to produce a pile of particles below. The shape of the pile is studied and $Q_K$ is proportional to the pile angle.

**TG3: Rotating drum** A rotating drum is filled to half with particles. The velocity profile of the grains is studied and $Q_K$ is computed from it.

**TG4: Blade test** A bucket is filled with particles. The particles are pushed with a blade driven at constant velocity. Force on the blade is studied and $Q_F$ is calculated from this. $Q_K$ is calculated from the total displacement of particles.
Chapter 5

Results

5.1 TE1: Collision test

Two spheres are shot against each other with the same velocity, see Figure 5.1. The test is made to see how the overlaps are handled from one timestep to another.

Gravity is turned off. Velocities are configured to produce an impact overlap of 1% and 10% of the particle radius. This corresponds to initial velocities of $v_1 = 0.01 m/s$ and $v_2 = 0.1 m/s$, respectively. The impact inclination is 45 degrees.

To compare the data between softwares, we look at trajectories and energies.
The collisions between granules are strongly dissipative, and this energy loss characterize kinematics of the total simulation. We look at the energy difference before and after impact. The smooth simulation software LIGGGHTS is run with the same timestep size as the other non-smooth ones just to show that it fails badly. It is also run with a smaller timestep (HQ). The difference in energy before and after impact is measured. See Table 5.1. As in the table above, you can see that the force dissipation is very small in Agx, LMGC90 and Bullet Physics. In LIGGGHTS, the simulation seems to blow up because of the relatively big timestep, hence we get a big energy difference. LIGGGHTS will therefore get a force quality value of zero and the others one.

$Q_E$ will be the energy average per particle averaged over the whole simulation.

### Trajectories

By looking at the trajectories, we can get a grasp about how the softwares handle collision friction and transfer of momentum. A small trajectory deviation after the collision tells us that the effective friction in the collision is small, while a large deviation will prove the opposite. Note that we are using the same friction coefficient in all softwares, according to Table 4.1. The particle trajectories are shown in Figure 5.2.

<table>
<thead>
<tr>
<th>Software</th>
<th>Overlap</th>
<th>$\Delta E$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Agx</td>
<td>0.01</td>
<td>$2.6855 \cdot 10^{-9}$</td>
</tr>
<tr>
<td>Bullet</td>
<td>0.01</td>
<td>$2.5750 \cdot 10^{-9}$</td>
</tr>
<tr>
<td>LIGGGHTS</td>
<td>0.01</td>
<td>$-2.9829 \cdot 10^{0}$</td>
</tr>
<tr>
<td>LIGGGHTS, HQ</td>
<td>0.01</td>
<td>$2.6706 \cdot 10^{-9}$</td>
</tr>
<tr>
<td>LMGC90</td>
<td>0.01</td>
<td>$2.6885 \cdot 10^{-9}$</td>
</tr>
<tr>
<td>Agx</td>
<td>0.1</td>
<td>$2.6764 \cdot 10^{-7}$</td>
</tr>
<tr>
<td>Bullet</td>
<td>0.1</td>
<td>$2.5435 \cdot 10^{-7}$</td>
</tr>
<tr>
<td>LIGGGHTS</td>
<td>0.1</td>
<td>-</td>
</tr>
<tr>
<td>LIGGGHTS, HQ</td>
<td>0.1</td>
<td>$2.6668 \cdot 10^{-7}$</td>
</tr>
<tr>
<td>LMGC90</td>
<td>0.1</td>
<td>$2.6905 \cdot 10^{-7}$</td>
</tr>
</tbody>
</table>

Table 5.1: Energy difference before and after impact, TE1.
The kinetic quality $Q_K$ will be estimated by comparing with high quality simulation data since we do not have any experimental data. The deviation in the trajectory from the high quality data in degrees is measured. The mean deviation $\Delta \phi$ in the two different tests will be used to calculate the value of $Q_K$. We set $Q_K = 1 - \Delta \phi/20$.

\footnote{Simulation done with a timestep of $10^{-5}$ in \textit{LIGGGHTS}.}

Figure 5.2: Particle trajectories in the collision test.
Table 5.2: Quality estimation from TE1

5.2 TE2: Shock propagation test

A number of spheres are lined up with no space in between them. One free particle is shot onto the leftmost particle in the direction along the line. See Figure 5.3.

The force propagation time and movement is compared. The number of particles is $N = 10$ and the velocity of the initially moving particle is $0.01 m/s$.

The restitution was set to zero in this test.
The estimation of motion quality was hard in this case; the softwares are all behaving differently in this simulation. See Figure 5.4. The behaviour is known and can be explained more deeply, but this explanation is beyond the scope of this investigation.

We will look at the force propagation in this line of spheres instead. When does the last sphere in the line “feel” the propagated force? In reality, the force would propagate with the speed of sound in the current material from the first sphere to the last. This is not what we expect from these simulations, due to the relatively large timestep we use. Instead, we look which software is closest to the real-world result.

A non-smooth algorithm should propagate the force in one timestep if the number of solver iterations is large enough. If the number of spheres is $N$, then $N$ iterations would be needed if the iteration order is from the first sphere and along the line. If the iteration order is random, more iterations will be needed. We will run this text with our standard value for number of iterations, $n_{\text{iter}} = 10$.

The difference between the non-smooth softwares is interesting, but hard to explain.
In the simulations, it took 4 timesteps for AgX to propagate the force. See Table 5.3. In Bullet, this took 3 timesteps. In LIGGGHTS, we are running the simulation with a much smaller timestep size so it’s hard to even compete with it when it comes to force propagation. The most surprising result comes from LMGC90. It instantaneously (in one timestep) propagates the force through the spheres at the same timestep size as Bullet Physics and AgX. We can conclude that LMGC90 is using some other strategy when solving for contacts and propagating the force.

The $Q_F$ value is set to $1 - 10^{t_{\text{propagation}}}$, where $t_{\text{propagation}}$ is the propagation time.

The energy quality is measured by looking at the energy difference between the LIGGGHTS high quality solution after all spheres collided and separated. The energy quality value will be calculated as $1 - (\Delta E_{\text{LIGGGHTS}} - \Delta E)/3 \cdot 10^{-8}$.

The time quality is as in the other tests as mentioned earlier. One interesting note is that all the softwares made the simulation faster than real time, and LIGGGHTS is still “in the game” although the small timestep.

<table>
<thead>
<tr>
<th>Software</th>
<th>$Q_K$</th>
<th>$Q_F$</th>
<th>$Q_E$</th>
<th>$Q_C$</th>
<th>$Q_M$</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>AgX</td>
<td>-</td>
<td>0.6</td>
<td>0.1</td>
<td>0.20</td>
<td>6.0</td>
<td>-</td>
</tr>
<tr>
<td>Bullet</td>
<td>-</td>
<td>0.7</td>
<td>0.7</td>
<td>0.05</td>
<td>0.04</td>
<td>-</td>
</tr>
<tr>
<td>LMGC90</td>
<td>-</td>
<td>1.0</td>
<td>0.1</td>
<td>0.93</td>
<td>-0.13</td>
<td>-</td>
</tr>
<tr>
<td>LIGGGHTS</td>
<td>-</td>
<td>1.0</td>
<td>1.0</td>
<td>0.22</td>
<td>1.5</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 5.4: Quality estimation in TE2
5.3 TE3: Column compression test

The spheres are stacked in a one dimensional pile under exertion of gravity and the compression is studied. See Figure 5.5. The particle positions are locked to a line and cannot move outside it. The number of particles is 30. To compare data, we look at total compression size.

The theory (Section 3.5) suggests that we should have a timestep smaller than $\sqrt{\frac{r}{2g}} = 0.0159$, which implies that a timestep $\Delta t = 0.01$ is small enough for the non-smooth simulation.

The number of Gauss-Seidel iterations $n_{\text{iter}}$ in the non-smooth softwares, is a parameter that affects the result significantly. $Y$ is also a parameter that affects the result drastically. We will therefore look at results with different values of these parameters.
The kinematic quality, $Q_K$, is determined by looking at the movement of the particles. What we want in this scene is a static pile of particles and the results is compared to a high quality LIGGGHTS simulation results. To compare this, we look at the mean kinetic energy $\langle E_k \rangle_t$. The $Q_K$ value is calculated with $Q_K = 1 - \langle E_k \rangle_t \cdot 3 \cdot 10^3$ to get values between zero and unity.

The force quality $Q_F$ is calculated by looking at the total compression of the column (Figure 5.6), and $Q_F$ will be calculated with $Q_F = \frac{h_t}{h_0}$.

The results are shown in Table 5.5. We can see that the smooth software LIGGGHTS produces an almost rigid stack of spheres while the non-smooth softwares produced a more compressed stack. The compression of the stack
Figure 5.7: Simulation done with a smaller value of Young’s modulus, $Y = 10^8$ (should give more elastic material).

decreases when increasing the number of contact iterations, and at 500 iterations the non-smooth softwares gives about the same result.

One can also note that the computing time $Q_T$ decreases for AgX and Bullet Physics when increasing the number of contact solving iterations. This happens because their iterative solver actually gets relieved. Less contact penetration left from the previous timestep will help the next solve to converge, and fast convergence means that the software is “done” faster.

When dealing with this kind of static systems, non-smooth softwares can be convenient to use. This is because the contact iteration number can be tweaked to give satisfying results. In the smooth approach, we need to change the time step size to increase contact quality.

These simulations were made with $Y = 10^9$, but simulations with $Y = 10^8$ were also made as shown in Figure 5.7. By comparing to Figure 5.6 (c), we can see that AgX is the only software that respond to a change in $Y$. 

Stefan Hedman, steffe@steffe.se 36 October 17, 2011
Table 5.5: Quality estimation in TE3. Note that LIGGGHTS, the smooth software, does not have any value for $n_{iter}$ since smooth softwares does not have a contact solver.

### 5.4 TE4: Pyramid stability test

![Pyramid stability test](image)

Ten spheres are stacked in a 2D pyramid shaped pile and the stability is studied. See Figure 5.8. Comparable data is the minimum timestep needed for a stable pyramid, if the pyramid is stable at all.

We use high friction and stiff material and vary the timestep to see when it is possible to build a stable pyramid. See Table 5.6.
Table 5.6: Largest timestep size when a pyramid of spheres does not collapse during a 2-second simulation.

\[
Q_K < 1 \cdot 10^{-7} < 1 \cdot 10^{-7} < 2.1 \cdot 10^{-3} < 4.5 \cdot 10^{-2}
\]

Table 5.7: Quality estimation of softwares in TE4

<table>
<thead>
<tr>
<th>Software</th>
<th>(Q_K)</th>
<th>(Q_F)</th>
<th>(Q_E)</th>
<th>(Q_C)</th>
<th>(Q_M)</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>AgX</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>16</td>
<td>-</td>
</tr>
<tr>
<td>Bullet</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-0.02</td>
<td>-</td>
</tr>
<tr>
<td>LMGC90</td>
<td>1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>58.8</td>
<td>-</td>
</tr>
<tr>
<td>LIGGGHTS</td>
<td>1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>9.36</td>
<td>-</td>
</tr>
</tbody>
</table>

5.5 TE5: Box on an inclined plane

Figure 5.9: “Box on an inclined plane” test. The box is replaced with four spherical particles in a tetrahedral shape to maintain the compatibility with all softwares.

A tetrahedron made of four composite spheres are placed on an inclined plane. See Figure 5.9. The reason why not using a real box with eight vertices are several. First, artefacts in the form of instability can occur when having more than three contacts against the same plane. Second, arbitrary dynamic polygons are not available in all softwares (such as LIGGGHTS and LMGC90).

The regular tetraedrons’ side is 3d and the spherical particles are mounted
with their centers of mass on the tetraedron vertices. The whole object is treated as one dynamic body where the spheres have a locked relative position to each other. I will further on refer to this object as a “box”.

The angle where the object starts to slide is measured to investigate the correctness of the friction in the physics software.

When the object starts to slide, we have $F_x = F_{fr} = \mu F_N = \mu mg \cos(\theta)$. We also have $F_x = mgsin(\theta)$, so $\mu = \tan(\theta)$. See Figure 5.10. An object with $\mu = 0.35$ should thus start to slide at the approximate angle $\theta = 19.3$ degrees.

Since the tetraedron is dropped onto the plane with a small velocity, it will slide a small distance or bounce on the plane before it stops or continues to slide. In the cases where there is small distinction between stopping and sliding, the sign of the acceleration determines the result.

The test is simple and straightforward. We will investigate the angle in which the box starts to slide.

<table>
<thead>
<tr>
<th>Software</th>
<th>18.0</th>
<th>18.1</th>
<th>19.0</th>
<th>19.2</th>
<th>19.3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bullet</td>
<td>-</td>
<td>Slide</td>
<td>Slide</td>
<td>Slide</td>
<td>Slide</td>
</tr>
<tr>
<td>AgX</td>
<td>-</td>
<td>-</td>
<td>Slide</td>
<td>Slide</td>
<td>Slide</td>
</tr>
<tr>
<td>LIGGGHTS</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Slide</td>
</tr>
<tr>
<td>LMGC90</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Slide</td>
</tr>
</tbody>
</table>

Table 5.8: Angles when the tetraedron starts to slide in the different softwares.

We will skip the quality values $Q_K$, $Q_F$ and $Q_E$ in this test and instead present the slide angles.
Spheres of diameter $d$ are poured into a cylinder container of radius $5d$. The number of particles $N$ is 2500. See Figure 5.11.

Comparable data is the force network and stress distribution along the cylinder axis. The vertical stress $\sigma_{zz}$ was investigated in [13], so we will do that in the same manner.

The pressure distribution for granular matter is quite different from the corresponding liquid version, since friction and support from the walls needs to be considered. For a liquid, the pressure depends linearly on the height, while the granular version is more evenly distributed. We will use the Janssen model as described in [13]. The stress is according to the Janssen model

$$\sigma_{zz}(z) = \rho g l \left[ 1 - \exp \left( -\frac{z_0 - z}{l} \right) \right]$$

(5.1)

where $l$ is the decay length $l = R/2\kappa\mu$ and $\kappa$ is the fraction of the weight carried by the side walls. See Figure 5.12.
Figure 5.12: The Janssen model. The pressure $\sigma_{zz}(z)$ is evenly distributed in a granular packing but near the top it decreases exponentially with the height $z$.

Figure 5.13: Vertical stress $\sigma_{zz}(z_i)$, as a function of $z_i$ in AgX (a), Bullet Physics (b), LIGGGHTS (c) and LMGC90 (d).

One can note that the particle compression is large in AgX and Bullet Physics, see Figure 5.14. This may be a result of too few solver iterations ($n_{\text{iter}} = 10$). Increasing the number of iterations, artefacts will submerge (see how variations of $n_{\text{iter}}$ affect stacking problems in TE3). However, the compression of the grains is interesting to compare in between the softwares.
To measure force quality, we look at the fit to the Janssen model. Clearly, \textit{LIGGGHTS} and \textit{LMGC90} produces a more evenly distributed pressure along the axis. \textit{AgX} and \textit{Bullet Physics} produces a result that is rather in the viscous regime. Therefore, we give the first two $Q_F = 1$ and the others $Q_F = 0$.

$Q_K$ is not considered as interesting in this test since the particles are static. Table 5.10 shows the performance of the different softwares.

<table>
<thead>
<tr>
<th>Software</th>
<th>$Q_K$</th>
<th>$Q_F$</th>
<th>$Q_E$</th>
<th>$Q_C$</th>
<th>$Q_M$</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>AgX</td>
<td>-</td>
<td>0.0</td>
<td>$4.2 \cdot 10^{-7}$</td>
<td>108</td>
<td>520</td>
<td>-</td>
</tr>
<tr>
<td>Bullet</td>
<td>-</td>
<td>0.0</td>
<td>$2.4 \cdot 10^{-7}$</td>
<td>3112</td>
<td>934</td>
<td>-</td>
</tr>
<tr>
<td>LMGC90</td>
<td>-</td>
<td>1.0</td>
<td>$2.3 \cdot 10^{-10}$</td>
<td>2575</td>
<td>254</td>
<td>-</td>
</tr>
<tr>
<td>LIGGGHTS</td>
<td>-</td>
<td>1.0</td>
<td>$2.9 \cdot 10^{-14}$</td>
<td>200</td>
<td>361</td>
<td>bin size=0.01</td>
</tr>
</tbody>
</table>

Table 5.10: Quality estimation of softwares in TG1.

### 5.7 TG2: Cylinder dam break

Particles in a cylindrical container without bottom is simulated until a state of rest, then the cylinder is raised and the particles will fall down on the ground and build a cone-shaped pile. See Figure 5.15. The number of particles $N$ is 5000, particle diameter is $d$, and the ground is constructed using a square grid layer of static spheres of diameter $3d$. 

Stefan Hedman, steffe@steffe.se
The procedure of the Cylinder dam break test. A cylinder is filled with $N$ particles, then the containing cylinder is raised at a constant velocity $v_{cyl}$. The particles forms a pile on the ground and the shape of it is measured.

The simulation result used for comparison is the inclination of the pile. In [9], the slope is about 20 degrees in a real-world experiment (the largest possible angle according to theory angle is about 24 degrees [20]). The experimental setup is characterized by the parameters $a = H/R$, where $H$ is the height of the filled silo and $R$ is the radius of the silo. In [9], one of the data sets had $a = H/R = 1$, $M = 1.6kg$, $R = 70.5mm$, $d = 350 \pm 50$ micrometer sandpaper substrate. The geometry dimensions and number of grains in this experiment is quite unsuitable for numerical simulation in our types of softwares. Therefore, the resulting inclination of the pile is assumed to be somewhat preserved or at least close to these result when using fewer, larger and spherical grains.

We set the particle size $d = 0.05m$ and $v_{cyl} = 1.6m/s$ in our tests. Since we are using spherical grains, we expect the pile inclination to be a bit smaller than real-world experiments.
For $Q_K$, we will look at the shape of the pile. See results in Figure 5.16. As described earlier, the inclination of the pile should be about 20 degrees. We set the $Q_K$ value to $1 - (20 - \phi)/20$.

![Figure 5.16: Pile in Bullet Physics, LMGC90, LIGGGHTS and AgX. Note that the simulation snapshots are not to scale relative to each other.](image)

<table>
<thead>
<tr>
<th>Software</th>
<th>$Q_K$</th>
<th>$Q_E$</th>
<th>$Q_F$</th>
<th>$Q_C$</th>
<th>$Q_M$</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>AgX</td>
<td>0.65</td>
<td>-</td>
<td>$6 \cdot 10^{-5}$</td>
<td>185</td>
<td>1110</td>
<td>-</td>
</tr>
<tr>
<td>Bullet</td>
<td>0.32</td>
<td>-</td>
<td>$2.3 \cdot 10^{-4}$</td>
<td>314</td>
<td>119</td>
<td>-</td>
</tr>
<tr>
<td>LMGC90</td>
<td>0.65</td>
<td>-</td>
<td>$4.7 \cdot 10^{-8}$</td>
<td>428</td>
<td>201</td>
<td>-</td>
</tr>
<tr>
<td>LIGGGHTS</td>
<td>0.80</td>
<td>-</td>
<td>$7 \cdot 10^{-5}$</td>
<td>539</td>
<td>39.7</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 5.11: Quality estimation of softwares in TG2
5.8 TG3: Rotating drum

A rotating drum of particles may be the hardest test for the physics softwares. It is also the most interesting to study when looking at granular simulations. Why? Because all three granular states are there: static (near the drum bottom), dense flow, and the gasous regime on top.

In [17], a rotating drum is filled to half with spherical particles. Velocity profiles $v_x(z)$ (where $e_x$ is along the flow direction and $e_y$ perpendicular to the drum rotation vector and $e_z$, see Figure 5.17) are studied.

Unfortunately, there was not enough time in the project to run the drum simulation in LMGC90.
What we will look at is velocity profiles. This will tell us about the flow of the particles and will depend on how the contact and friction solving is performing during the granular flow. The number of particles is $N = 2500$ and the drum rotational speed is $w = 3.0\text{rad/s}$. The dimensions of the drum is the length $L = 7d$ along the rotational axis, and the inner radius is $\sqrt{\frac{8Nd^3}{\pi w^2}} \approx 0.14m$. The choice of radius is because we want a half-filled drum. The granular mass shape in the different softwares is viewed in Figure 5.18.

To investigate the flow state of the simulation, the inertial number was measured. The distribution of the inertial number is presented in Figure 5.19. The inertial number is measured as mentioned in Section 3.2.5 during one timestep.
For further investigation of the flow, velocities are investigated and presented in a velocity profile plot as in Figure 5.20.
Figure 5.20: Velocity profile plot, gives information about volume compression and flow. \( z \) and \( v_x \) are defined as in Figure 5.17.

The inertial number will tell us the characteristics of the flow. The inertial number is calculated and averaged over 3 consecutive timesteps when we have reached a steady flow.

<table>
<thead>
<tr>
<th>Software</th>
<th>( \langle I \rangle )</th>
<th>( Q_K )</th>
<th>( Q_F )</th>
<th>( Q_E )</th>
<th>( Q_C )</th>
<th>( Q_M )</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>AgX</td>
<td>(4.7 \cdot 10^{-2})</td>
<td>-</td>
<td>-</td>
<td>(1.2 \cdot 10^{-3})</td>
<td>85</td>
<td>420</td>
<td>-</td>
</tr>
<tr>
<td>Bullet</td>
<td>(5.1 \cdot 10^{-2})</td>
<td>-</td>
<td>-</td>
<td>(1.3 \cdot 10^{-6})</td>
<td>468</td>
<td>2026</td>
<td>-</td>
</tr>
<tr>
<td>LMGC90</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>See [17].</td>
</tr>
<tr>
<td>LIGGGHTS</td>
<td>(7.0 \cdot 10^{-8})</td>
<td>-</td>
<td>-</td>
<td>(1.1 \cdot 10^{-6})</td>
<td>173</td>
<td>20.7</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 5.12: Quality estimation of softwares in TG3. Unfortunately, there was no time over to run this test in LMGC90.

The results from the drum simulations are shown in Table 5.12. *Bullet Physics* used an awful lot of memory in this test, and the reason is not clear. AgX was the fastest software and *LIGGGHTS* used least memory. From Figure 5.20, we can see that the motion of the grains in the *LIGGGHTS* simulation is quite different from the motion we see in AgX and Bullet Physics. The grains form an inner “core” of nearly static granular material while the grains near the surface and drum are moving faster. The case in the non-smooth softwares is not the same, the “core” is smaller and the particles are moving faster.
5.9 TG4: Blade test

A blade moves through a collection of grains. The total normal force on the blade is measured and compared with experimental data from [10]. The particle movement is also studied.

A box of size 6×3 m was filled with particles of density $\rho = 2.63 \cdot 10^3 \text{kg/m}^3$. A blade of size 0.1 × 3.0 × 1.0 m plowed through the grains at a cutting depth of about 1 m with a velocity of $v_{\text{blade}} = 1.0 \text{m/s}$. The diameter of the grains were adapted to get the container filled, and the resulting diameter $d$ was about 0.34 m. See Figure 5.21. The net force exerted onto the blade in the direction of movement is measured.

In our setup, we use $N = 2500$ particles and we use a kinematic blade. We measure the net force on the blade $F_{\text{blade}}$. The empirical model that we compare to is proposed by McKaye [18]. The total force on the blade is given by

$$P = \left( \rho_b g d_{\text{cut}}^2 K_p + \rho_b g v_{\text{blade}}^2 d_{\text{cut}} K_a + q d_{\text{cut}} K_q \right) \omega$$

(5.2)

with the bulk density $\rho_b$, gravity $g$, cutting depth $d_{\text{cut}}$, coefficient for lateral earth pressure $K_p$, cutting speed $v$, surcharge pressure $q$ and coefficients to account for the influence of speed $K_a$ and surcharge pressure $K_q$. 

Figure 5.21: The procedure of the Blade test. A is bucket filled with particles and a blade of a constant velocity $v_{\text{blade}} = 1.0 \text{m/s}$ plows through the grains. The force on the blade is investigated.
The horizontal force acting on the blade is given by

\[ H(x) = P(x)\sin(\alpha + \psi) + F_m(x) \]  

(5.3)

with the rake angle \( \alpha \) and the soil–tool friction \( \psi \). In order to account for the piling effects in front of the blade, an additional force \( F_m \) is introduced and the surcharge pressure \( p \) is increasing as the blade moves forward. As said in [10], these empirical formulas provide only a rough estimate of the expected forces on the blade. However, this helps to get an impression of whether or not the results of the simulations are reasonable. For the comparison of different numerical setups this is of no influence. Therefore, no further detail is given on the use of these formulas.

![Figure 5.22: Total normal force on the blade as a function of time in TG4.](image)

Unfortunately, there was not enough time for extracting the force on the blade in LMGC90 during the project. All other data is being analysed though.

As you can see in Figure 5.22, the force on the blade is relatively small in the Bullet Physics simulation. That is probably because the particles were densely packed and pressed together, and the cutting depth \( d_{cut} \) is therefore smaller. The compression of particles is discussed in Section 6.

We connect these results to the \( Q_F \) quality measurement. I want to stress that The McKaye model is a very rough estimate in our case, so setting this quality value is difficult. They all fit quite well to the model.

We also measure is the sum of particle displacements \( \Delta_x(t) \), according to Equation 5.4. This will give some hints on how strong particles are affected by each other in the force network connections originating from the blade. If
the performance from the contact and friction solver is better, then the total movement of particles will be bigger. Also, if particles flows back behind the blade fast, then $\Delta x(t)$ will become smaller.

$$\Delta x(t) = \frac{1}{N} \sum_{i=1}^{N} \|x_i(t) - x_i(0)\|$$

$$\text{(5.4)}$$

Figure 5.23: Average particle displacement as a function of time in TG4.

A “good” simulation software should handle the contacts and friction well. Accurate contact and friction handling implies a larger value of the particle displacement from the original position. $Q_K$ will therefore be a measure of the displacement, assuming the error lies in the friction handling. We take the last value of the displacement and map it to a linear scale between 0 and 1. See Figure 5.23.

The energy average $Q_E$ is per particle per time $t = [5.95, 6.00]$.

<table>
<thead>
<tr>
<th>Software</th>
<th>$Q_K$</th>
<th>$Q_F$</th>
<th>$Q_E$</th>
<th>$Q_C$</th>
<th>$Q_M$</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>AgX</td>
<td>0.65</td>
<td>0.8</td>
<td>$1.8 \cdot 10^{-4}$</td>
<td>74</td>
<td>359</td>
<td>-</td>
</tr>
<tr>
<td>Bullet</td>
<td>0.40</td>
<td>0.8</td>
<td>$4.2 \cdot 10^{-5}$</td>
<td>108</td>
<td>142</td>
<td>-</td>
</tr>
<tr>
<td>LMGC90</td>
<td>0.85</td>
<td>-</td>
<td>$5.5 \cdot 10^{-5}$</td>
<td>82</td>
<td>132</td>
<td>Force on blade not available.</td>
</tr>
<tr>
<td>LIGGGHTS</td>
<td>0.58</td>
<td>0.5</td>
<td>$3.5 \cdot 10^{-5}$</td>
<td>280</td>
<td>128</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 5.13: Quality estimation of softwares in TG4
Chapter 6

Conclusions

In this chapter, the quality features of the softwares are discussed, as well as the choice between smooth or non-smooth simulation approaches.

6.1 Quality features

6.1.1 Bullet Physics

The strength of Bullet Physics is its computational speed $Q_C$. It was definitely the fastest software in the elementary tests where it also didn’t use any lot of memory. It seems like it is more optimized for few and large objects, according to the manual objects of size between 0.1 and 5 meters [4].

Looking at the summation tables, one can conclude that Bullet Physics does not have good results for the kinematic quality $Q_K$, and not too good force quality $Q_F$ either.

When starting to do tests in Bullet, I noticed that the contact behaviour was strange. Sometimes the contacts were not processed by the solver at all, and ruined the results. By tweaking the solver parameter $gContactBreakingThreshold$, Bullet performed better for our simulations. It is a necessity to change this threshold when simulating small objects. The contact breaking threshold controls when a contact between two separating objects is not a contact any more. The default value fits for objects of a size between 0.1 and 5 meters [4]. To make the contact behaviour turn out right for small objects, $gContactBreakingThreshold$ needs to be set to a smaller value. Alternatively, scale up the objects in the simulation. The latter is easier since the source code does not need to be recompiled (several times since picking a good value for this threshold is not a straight forward thing).

Perhaps the bad results from Bullet Physics is because we chose the wrong value of $gContactBreakingThreshold$, or perhaps Bullet Physics is just not optimized for the small object scale that we have in our simulations.
Looking on the positive side, *Bullet Physics* can provide many other types of constraints than just contacts. It is therefore possible to construct e.g. robots or other complicated machines. This cannot be done in *LIGGGHTS* and *LMGC90*.

Choose *Bullet Physics* if you want fast, visually appealing simulations and do not really care about correct physics.

### 6.1.2 AgX Multiphysics

*AgX* and *Bullet Physics* are on the same scale of quality in the tests, although *AgX* seems to be better at physics ($Q_K$ and $Q_F$). *AgX* seems to be the only simulation software that responds to a change of Young’s modulus.

*AgX* was the fastest software in the granular tests, where it also used some more memory than the average.

Together with *Bullet Physics*, *AgX* had a characteristic compression of the granular particles compared to the other softwares. The compression can be seen in some of the graphs. It is probably there because the Gauss-Seidel contact solver don’t converge in many of the cases, in lack of solver iterations. Doing the same tests with more contact solver iterations $n_{iter}$ would decrease the artefact size (according to theory) - but at the same time increase the calculation time. The reason why *LMGC90* converges better in the same situations may be because it uses an other type of solver that works differently.

Similar to *Bullet Physics*, *AgX* can provide many other types of constraints than just contacts. It is not clear to me whether *Bullet Physics* or *AgX* has got the richest set of constraint types but they are far beyond *LIGGGHTS* and *LMGC90*.

Overall, *AgX* did not produce the best results, neither the worst. It was not the fastest software, nor the slowest. There are no apparent disadvantages with using *AgX*, but the fact that it a linear Gauss-Seidel solver makes it hard to simulate systems such as the pyramid test.

### 6.1.3 LMGC90

*LMGC90* was remarkably well performing in many of the tests the average value of $Q_K$ is better than the other two non-smooth softwares. $Q_F$ is also large on average, which shows that this software is good at physics. It does not use remarkably much memory either; in the granular test it used less memory than at least one of the non-smooth softwares (the smooth software was best).

*LMGC90* is not computationally fast relative to the other non-smooth softwares. In the silo test it took 25 times as much time as the fastest software, and it wasn’t the fastest physics engine in any of the tests. It
performed better when simulating liquid-like granular matter rather than static.

Also, LMGC90 does only have support for a limited variety of shapes and constraints which makes it hard to use when simulating e.g. machines.

6.1.4 LIGGGHTS

LIGGGHTS was definitely the slowest physics simulation software, but that is mostly the result from choosing a small timestep (which is required by smooth methods). This makes LIGGGHTS less suitable for real time simulation.

In LIGGGHTS, only a limited number of dynamic object shapes could be used. It is more than enough for granular simulations but not other systems.

The memory usage by LIGGGHTS was the smallest in the granular tests compared to the other softwares and this can be improved further by tweaking the collision detection parameters. For example, LIGGGHTS only used 20 megabytes for the drum simulation (with 2500 particles) while the others used more than 420.

One can also note that LIGGGHTS needed more memory and computational time for the static silo compared to the drum with the more flowing state.

LIGGGHTS confirms the theory of smooth simulation software in terms of memory and computational speed. The physics is also of high quality. Choose LIGGGHTS if you do not aim for real time simulation, need a memory cheap tool and want high quality results.

6.2 Smooth or non-smooth?

The key question was from the beginning which approach to use to simulate granular material. I will in this section try to enlight those who are facing this question.

Type of problem Look at the efficiency domains plot in Figure 3.7. What domain is your simulation in? By choosing the most suitable approach, efficiency can be gained.

Implementation Implementation is easier and more straight forward in the smooth approach. If you are making an own simulation tool, you might save implementation time here.

Support for more types of constraints The non-smooth simulation tools does often have support for more types of constraints than just contact type ones. This does mostly result in a more generic software which...
can handle more complicated structures such as machines interacting with granular material.

**Deformability** The smooth approach assumes elastic and deformable material. To get very stiff material, one has got to use a very small timestep or switch to the non-smooth approach. The non-smooth softwares got the advantage to be able to simulate both rigid and elastic contacts as discussed in Section 3.4.1.

**Speed** If you aim for fast simulation (or even real time), choose the non-smooth approach. When simulating large systems with many contacts, the smooth approach may be faster.

**Memory usage** For large complicated systems with many contacts, the non-smooth approach may consume more memory. If your hardware has got a limited amount of memory, the smooth approach may be a better choice.

### 6.3 Pipeline framework

Most time in this project was put on developing the testing framework and maintain the compatibility with the four simulation softwares. New bugs were found constantly and therefore simulation data had to be regenerated many times.

The development of the pipeline took away the focus from the theory part and forced me to extend the project time. For a similar future project, I would recommend to use som existing testing framework like PAL [21], though it does not contain bindings to either of *LIGGGHTS*, *AgX* or *LMGC90* that we wanted to investigate in this project.
Bibliography


[16] LMGC90. lmgc.univ-montp2.fr/dubois/LMGC90.


Thanks to

Martin Servin, supervisor For the best supervising and guidance.

Frédéric Dubois, developer of LMGC90 For personal help on using the simulation software.
Christoph Kloss, developer of LIGGGHTS For all help and constant presence in the LIGGGHTS forum.

Erwin Coumans, developer of Bullet Physics