Thesis for the Degree of Master of Science in Engineering Physics

# Parallel projected Gauss-Seidel solver for large-scale granular matter

Examining the physics of the parallel solver and development of a multigrid solver

**Johan Sundberg**
**February 6, 2014**

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

Department of Physics
Faculty of Science and Technology
Umeå University
Umeå, Sweden

**Abstract**

Granular matter is found everywhere in nature and some examples include sand, rice, coffee beans and iron ore pellets. Many different methods exists for simulating granular matters using computers. In the scope of this thesis a physics engine called AgX Dynamics from Algoryx Simulation AB is used to investigate and further develop methods involving the discrete element method. During the first half of 2013 a parallel solver for the projected Gauss-Seidel method was implemented in AgX in order to speed up the simulation time of simulations involving granular materials. In this thesis project it is shown that the behaviour of the physics of this parallel solver is identical to the serial solver. Secondly this thesis works on the development of a multigrid solver for the Gauss-Seidel method. Multigrid in this context means that the particle system is partitioned in space. Each partition is then merged into a rigid body and contact forces between these rigid bodies is solved to machine precision using a direct solver. The forces from this direct solve is then used when solving the internal part of the partitions using an iterative projected Gauss-Seidel method. The motivation for developing a multigrid method is to achieve faster convergence and even more speed-up of the solver. Numerical experiments has been performed on a 1D column and a 3D silo. The results show high potential of the method and the one-dimensional column behaves closer to a direct solver than an iterative solver.

The thesis was done for UMIT Research Lab, Umeå University and Algoryx Simulation AB.

**Sammanfattning**

Granulärt material återfinns överallt i vår natur och i industrin; några exempel är sand, ris, kaffebönor och järnmalmspellets. Det finns många olika metoder för att simulera granulära material m.h.a datorer. I det här examensarbetet så används en fysikmotor, AgX Dynamics, från Algoryx Simulation AB för att undersöka och vidareutveckla metoder som involverar diskreta elementmetoden. Under första halvan av 2013 utvecklades en parallel version av den numeriska lösaren projected Gauss-Seidel i fysikmotorn AgX. Syftet var att simuleringar av partikelsystem skulle kunna köras ännu snabbare. I denna rapport så visas att fysiken i den parallela lösaren beter sig identiskt med den seriella numeriska lösaren. Vidare så påbörjas arbetet på utvecklingen av en multigrid lösare för Gauss-Seidel metoden. Med multigrid menas att partikelsystemet i simuleringen delas upp i partitioner i rymden. Av varje partition skapas sedan en stelkropp av alla partiklar i partitionen och egenskaper för den stela kroppen räknas ut ifrån partikelsystemet i partitionen. Kontaktkrafterna mellan dessa stela kroppar löses sedan med en direkt lösare. Krafterna från den direkta lösningen används för att simulera partiklarna internt i en partition med en iterativ projected Gauss-Seidel metod. Motiveringen för att utveckla en multigrid metod är att uppnå snabbare konvergens i simuleringen och ännu snabbare körningar. Numeriska experiment har utförts för en 1D kolumn och en 3D cylinder. Resultaten för en enkel 1D simulering visar att metoden har potential och den beter sig närmare en direkt lösare än en iterativ lösare.

**Acknowledgments**

There are many people i would like to thank that helped me during my work on this thesis. First of all Martin Servin, my supervisor who always gave constructive feedback on my work and helped develop the ideas for the thesis.

Claude Lacoursière has been a great help with all sorts of things from linear algebra to advanced numerical methods.

Da Wang, who together with Martin Servin has explained the physics behind the granular simulations to me. Many characters have been sent back and forth between me and Da on Skype, thanks for all the help.

The guys at Algoryx; Nils Hjelte, Tomas Berglund, Fredrik Nordfelth, Niklas Melin and everyone else has helped me a great deal with the inner workings of the physics engine and how to set up scenes and extract simulation data.

# Contents

# Chapter 1

# Introduction

## 1.1 Background

The Gauss-Seidel method has been used for a long time to iteratively solve systems of linear equations. It is widely used within many areas of computational science. In the context of this thesis, a projected Gauss-Seidel (PGS) method is used to solve a Mixed Linear Complementarity Problem (MLCP) that are used to calculate the forces and velocity updates in the physics engine AgX Dynamics (AgX) from Algoryx Simulation AB[1]. This particular project focuses on the simulation of granular materials.

Granular materials includes a large group of materials found in nature, including for example coal, sand, rice, coffee beans, ball bearings and iron ore pellets. This material is found everywhere in nature and is the second most manipulated material in industry, after water [10]. This makes it of great interest for scientists to study and simulate granular materials.

Today most processors for laptops and desktop computers are so-called multi-core processors. This means that they have more than one central processing unit, CPU, on the chip. It is not uncommon to have anywhere between 2-8 cores on a modern laptop or desktop computer today. Furthermore, todays processors are not becoming faster in terms of processing power per core, instead the manufacturers of processors are adding more cores to the processor. The consequence of this is that if you cannot parallelise your problem to make it run on more than one core at the same time, you cannot make your computer program run any faster; hence it is of great interest for the industry to develop algorithms and code that can run in parallel on many cores at the same time to speed up the execution of simulations.

Before work was started on this thesis it was not known if the physics of the parallel implementation in AgX differed from the serial implementation and if so by how much. It was suspected that they would differ some and the reason for this is as follows. The parallel solver divides up the particles in space and solves them in a certain order conceptualized by the solve graph in figure 2.7. From this follows that the different sections are solved in different orders because of threads executing and completing at different times compared to the serial solver. This means that the ordering of the equations in the iteration matrix (equation 2.10) are different from the parallel and the serial solver, and it should affect the simulation.

## 1.2 Purpose and objectives

Algoryx Simulation AB[1] has developed a parallel version of their projected Gauss-Seidel solver. This master thesis has two goals.

- Investigate the physics of the parallel solver and examine if and how much it differs from the serial version.
- Develop a multigrid solver for the discrete element method for granular media simulations.

In order to accomplish this a fair amount of work had to be put into the design and development of a simulation and post-processing pipeline that could be used to effectively compare the physics of two or more different simulation runs. The motivation to develop a multigrid method is to achieve faster convergence and even more speed-up of the simulations.

## 1.3 Notations

Throughout this thesis the words projected Gauss-Seidel and parallel projected Gauss-Seidel will be used extensively, so these will be refered to as PGS and PPGS respectively. The discrete element method will be referred to as DEM and the non-smooth discrete element method as NDEM. AgX Dynamics will often be abbreviated to AgX.

Vectors in this thesis will have a boldface font and a bar above it, for example $\bar{\mathbf{r}}$. Matrices will also have a boldface font.

# Chapter 2

# Theory

The theory chapter first describes what granular matter is and how it is simulated using non-smooth DEM. After this a description and motivation for projected Gauss-Seidel is given. The way projected Gauss-Seidel is parallelized is described as well as an overview and algorithm of a multigrid DEM solver.

## 2.1 Granular matter using non-smooth DEM

Granular matter can be defined as follows[2]. Granular medium is a collection of rigid macroscopic particles. The size of the particles are above 100µm, with size meaning the diameter if it is a sphere or spherical-shaped object. The consequences of this limit to particle size is that the interactions of the granules can be modeled by coloumb friction and the hertz model, while for example if the particles are of size $\sim$ 1-100µm then long range molecular force models such as van der Waals forces must be used. Cohesion and air drag also has to be taken into account when below the granular size limit in order to get correct physics.

In the scope of this thesis the granular media is simulated using the physics engine AgX [1] and the non-smooth DEM method. For a comparison between smooth and non-smooth DEM, see the paper by Servin et. al [12]. A short summary of how granular matter is simulated in AgX is described below, while more in-depth material can be found by reading the Phd thesis by Lacoursiere, C [7].

In the non-smooth DEM method (NDEM) the velocity in the simulation may change discontinuously in time. It considers impacts and the stick-slip friction transitions to occur as instantaneous events. Continuous contacts are modeled by non-penetration constraints. The rigid bodies in the simulation are modeled using the Newton-Euler equations of motion.

$$M\frac{dv}{dt} = -\nabla U(x) + F(x, v, t) + G^T\lambda, \tag{2.1}$$

where $v = \frac{dx}{dt}$ is the velocity, $M$ is mass, $U$ potential, $F$ is external forces and $G^T\lambda$ are constraint forces with lagrange multiplier $\lambda$ and Jacobian $G$.

Particle overlap is denoted by $g(x)$ and the non-penetration constraints are $g(x) \geq 0$. The constraint force acts to maintain this condition throughout the simulation. The Jacobian $G = \frac{\partial g}{\partial x}$ determines the direction of the constraint force and the Lagrange multiplier ($\lambda$) the magnitude. The Lagrange multiplier becomes an auxiliary variable to v that also needs to be computed numerically.

As a consequence of $g \geq 0$, the contact constraint force ($\mathbf{f}_n = G^T\lambda; \mathbf{f}_n$ being in the normal direction) and the normal relative contact velocity ($\mathbf{u}_n$) should satisfy the Signorini condition:

$$\begin{aligned} \mathbf{f}_n &\geq 0 \\ \mathbf{u}_n &\geq 0 \\ \mathbf{f}_n \cdot \mathbf{u}_n &= 0 \end{aligned} \tag{2.2}$$

Similarly, friction can be modeled as a velocity constraint, $G_t v = 0$, meaning that the relative tangential contact velocity should vanish. For this an additional multiplier $\lambda_t$ is introduced. The frictional constraint force $f_t = G_t^T \lambda_t$ should satisfy the Coloumb law of friction $|F_t| \leq \mu |F_n|$ which in this case translates to $|\lambda_t| \leq \mu |\lambda_n|$, where $\mu$ is the friction coefficient and $G_t$, $G_n$ are assumed normalized.

Using the SPOOK[7] algorithm for numerical integration of the constrained equations of motion the stepping scheme in equation 2.3 is aquired. Here $n$ is discretized time so that $n$ is the current time step and $n+1$ the next one.

$$\begin{aligned} \mathbf{x}_{n+1} &= \mathbf{x}_n + h\mathbf{v}_{n+1} \\ (\mathbf{v}_{n+1}, \lambda_{n+1}) &= MLCP(\mathbf{x}_n, \mathbf{v}_n, \mathbf{H}, \mathbf{b}) \end{aligned} \tag{2.3}$$

where MLCP is an abbreviation of mixed linear complementarity problem, a special case of a linear complementarity problem (LCP). A linear complementarity problem is an optimization problem that originates from mathematical optimization theory in which there are slack variables and complementarity conditions. MLCP extends LCP to include free variables. The MLCP to be solved each time step is:

$$\begin{aligned} \mathbf{Hz} + \mathbf{b} &= \mathbf{w}_+ - \mathbf{w}_- \\ 0 \leq \mathbf{z} - \mathbf{l} &\perp \mathbf{w}_+ \geq 0 \\ 0 \leq \mathbf{u} - \mathbf{z} &\perp \mathbf{w}_- \geq 0 \end{aligned} \tag{2.4}$$

where $\mathbf{l}$ and $\mathbf{u}$ are the upper and lower limits allowed in the solution $\mathbf{z}$, $\mathbf{w}_+$ and $\mathbf{w}_-$ are slack variables and

$$\mathbf{H} = \begin{bmatrix} \mathbf{M} & -\mathbf{G}_t^T & -\mathbf{G}_n^T \\ \mathbf{G}_t & \mathbf{\Gamma} & 0 \\ \mathbf{G}_n & 0 & \mathbf{\Sigma} \end{bmatrix}, z = \begin{bmatrix} \mathbf{v}^{i+1} \\ \lambda_t^{i+1} \\ \lambda_n^{i+1} \end{bmatrix}, b = \begin{bmatrix} -\mathbf{M}\mathbf{v}_n - h\mathbf{F} \\ \frac{4}{h}\gamma g - \gamma \mathbf{G}_n \mathbf{v}_i \\ 0 \end{bmatrix} \tag{2.5}$$

The terms $\mathbf{\Gamma}$, $\mathbf{\Sigma}$ and $\gamma$ are constraint regularization and stabilization terms. These introduce some elasticity and damping to the system in order to make it easier to solve numerically. It is possible to formulate the constraints such that the constraint force and overlap magnitude $g(x)$ follows the Hertz contact law and that regularization may be mapped to standard material parameters such as Young's modulus.

The MLCP can be understood as a linear system with inequality conditions for the solution variable, in this case $(\mathbf{v}_{n+1}, \lambda_{n+1})$. The inequality conditions comes from the Signorini and Coloumb laws. MLCP's are harder to solve than linear systems, PGS being one of the few algorithms that works in practice for large MLCP's of the type that occur for NDEM.

One of the properties studied in this thesis is the contact overlap between two particles, visually described in figure 2.1.

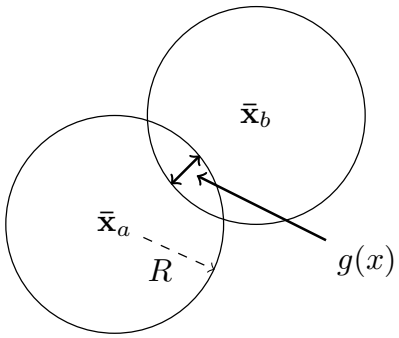



Figure 2.1: The overlap, $g(x)$, of two spheres in DEM

The contact overlap for two identical spheres is calculated as:

$$g(x) = 2 \cdot R - |\bar{\mathbf{x}}_a - \bar{\mathbf{x}}_b|, \tag{2.6}$$

where $R$ is the particle diameter and $\bar{\mathbf{x}}_a$, $\bar{\mathbf{x}}_b$ being the center position of the two particles. One interesting aspect of granular matter is that force networks are formed inside the material[2]. Examples of these force networks can be seen in figure 2.2 and 2.3 below. Within the force network in the granular material, both strong and weak chains are formed. This is a very important property of granular material which gives rise to some special physics. One effect is that jamming occurs, an example of this is sand in a hourglass where sand sometimes jams in the narrowest part of the hourglass. Another interesting effect is the so-called Janssen effect which is the reason why silos that hold grain or iron ore pellets do not break[2]. Strong chains are formed on the inside walls of the silo which helps to carry the load, unlike a water pillar in which the pressure increases linearly as the height increases. Examples of this from the simulations performed in this thesis can be seen in figure 2.2 for a 3d cylinder and figure 2.3 for a rotating drum. Each line in the force network examples is a contact between two granules and the intensity of the color is the normal force magnitude for that contact.

## 2.2 The Gauss-Seidel method

There is a category of numerical solvers called stationary iterative methods which are used to iteratively solve systems of linear equations. Linear stationary iterative methods are also called relaxation methods because in a way one can say that they relax into the correct solution by measuring the error between each iteration and correcting the solution. The problem of iteratively finding the solution to a linear system of equations is described mathematically in the following way[6]:

Find $x$ belonging to $\mathbb{R}^n$ such that

$$\mathbf{Ax} = \mathbf{b}, \tag{2.7}$$

where $\mathbf{A}$ is an $n \times n$ real matrix and $\mathbf{b}$ is a $n \times 1$ vector. One can reformulate this equation into a linear fixed-point iteration scheme by first rewriting $\mathbf{Ax} = \mathbf{b}$ into equation 2.8. See appendix A.1 how to do this rewriting (I is the identity matrix).

$$\mathbf{x} = (\mathbf{I} - \mathbf{A})\mathbf{x} + \mathbf{b} \tag{2.8}$$

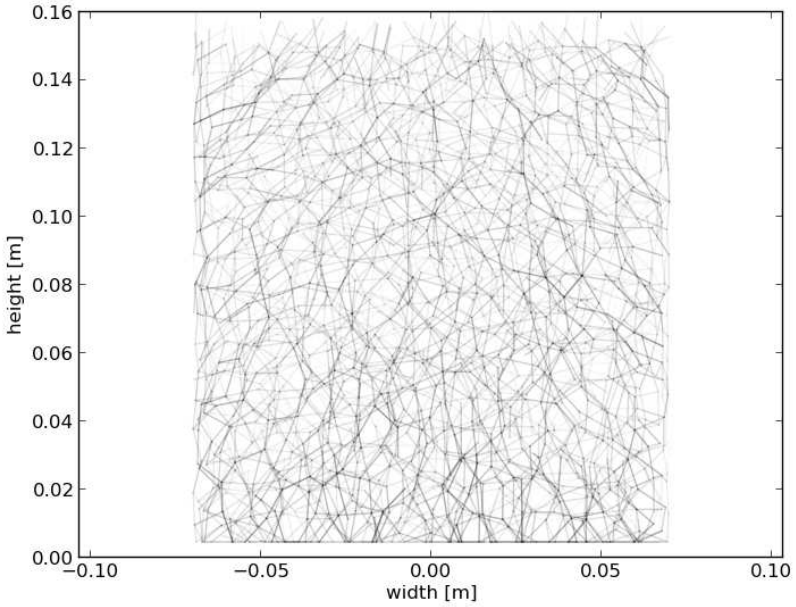



Figure 2.2: Force network in a 3d cylinder containing 30,000 particles and simulated using 500 iterations. The height is clipped at 15 particle diameters.

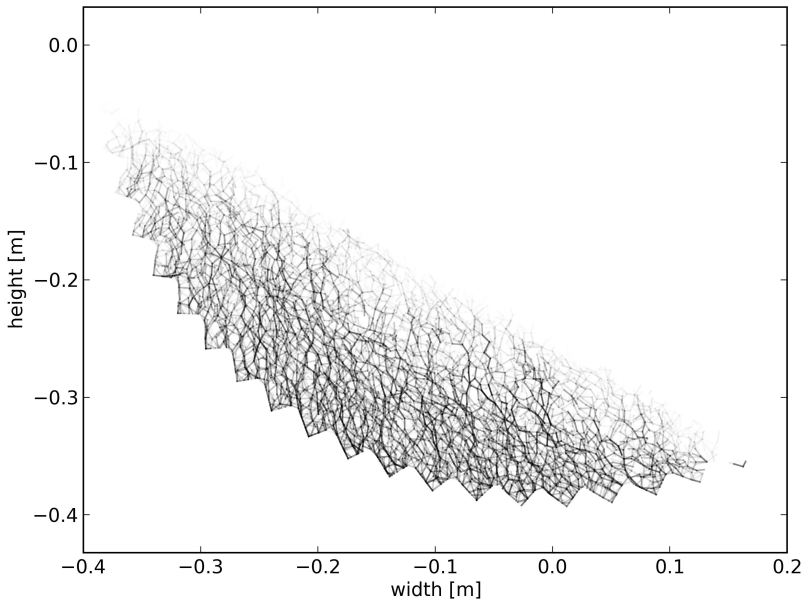



Figure 2.3: Force network in a rotating drum containing 25,000 particles and simulated using 500 iterations

From this the so-called Richardson iteration is defined as such:

$$\mathbf{x}_{k+1} = (\mathbf{I} - \mathbf{A})\mathbf{x}_k + \mathbf{b} \tag{2.9}$$

A more general form of this equation, where other matrices than the identity matrix can be used, looks like:

$$\mathbf{x}_{k+1} = \mathbf{M}\mathbf{x}_k + \mathbf{c}, \tag{2.10}$$

where $\mathbf{M}$ is an $n \times n$ matrix, same size as $\mathbf{A}$, called the iteration matrix. This matrix can be constructed in a number of different ways, which gives rise to the different varieties of stationary iterative solvers. Some common ones include the Jacobi, Gauss-Seidel and successive overrelaxation (SOR) methods. These three methods uses the fact that you can construct matrix M in more and less clever ways to improve the iteration scheme.

The Gauss-Seidel method uses a matrix splitting in the following way:

$$\mathbf{A} = \mathbf{L} + \mathbf{D} + \mathbf{U}. \tag{2.11}$$

Applying this to equation 2.7 and rearranging: $(\mathbf{L}+\mathbf{D}+\mathbf{U})\mathbf{x} = \mathbf{b} \implies (\mathbf{L}+\mathbf{D})\mathbf{x} = \mathbf{b} - \mathbf{U}\mathbf{x} \implies \mathbf{x} = (\mathbf{L}+\mathbf{D})^{-1}\mathbf{b} - (\mathbf{L}+\mathbf{D})^{-1}\mathbf{U}\mathbf{x}$. When applying the Richardson iteration (equation 2.9) this becomes:

$$\mathbf{x}_{k+1} = (\mathbf{L}+\mathbf{D})^{-1}\mathbf{b} - (\mathbf{L}+\mathbf{D})^{-1}\mathbf{U}\mathbf{x}_k \tag{2.12}$$

Considering equation (2.10) the iteration matrix $\mathbf{M}$ becomes $\mathbf{M} = (\mathbf{L}+\mathbf{D})^{-1}\mathbf{U}$ and $\mathbf{c}$ in the same equation is $\mathbf{c} = (\mathbf{L}+\mathbf{D})^{-1}\mathbf{b}$, and this is the iteration scheme that the Gauss-Seidel method uses.

### 2.2.1 Convergence of Gauss-Seidel

If $\sigma(\mathbf{A})$ is defined as the set of eigenvalues $\lambda$ of $\mathbf{A}$ then the spectral radius $\rho(\mathbf{A})$ is defined like so:

$$\rho(\mathbf{A}) = \max_{\lambda \in \sigma(\mathbf{A})} |\lambda| \tag{2.13}$$

That is, the spectral radius of $\mathbf{A}$ is the absolute value of the largest eigenvalue of $\mathbf{A}$.

Considering the general iteration scheme in equation (2.10), according to the book by Kelley (theorem 1.3.1, page 7, Kelley[6]), one of the sufficient conditions for convergence is that $\rho(\mathbf{M}) < 1$, where $\mathbf{M}$ is the iteration matrix and $\rho(\mathbf{M})$ is the spectral radius of $\mathbf{M}$.

Furthermore, another sufficient condition for the convergence of the Gauss-Seidel method is that the matrix $\mathbf{A}$ is strictly diagonally dominant or an irreducibly diagonally dominant matrix. (theorem 4.9, end of page 121, Saad[11]).

To conclude the convergence results, Gauss-Seidel will converge if one of these two criteria is fulfilled, considering the matrix $\mathbf{A}$ and iteration matrix $\mathbf{M}$

- $\rho(\mathbf{M}) < 1$
- $\mathbf{A}$ is strictly or irreducibly diagonally dominant

The rate of convergence for Gauss-Seidel is not fast. A figure from the paper by Servin et. al[12] is shown in figure 2.4 below to illustrate this.

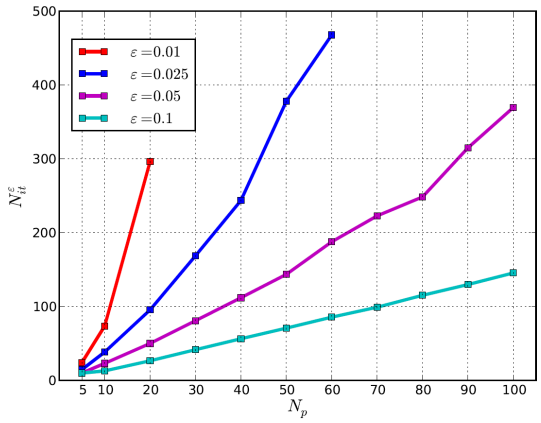



Figure 2.4: The required number of iterations depending on error and number of particles in a 1d column. Image taken from reference[12].

Here $\epsilon$ is error tolerance, $N_{it}$ is number of required iterations to achieve specified error tolerance and $N_p$ is number of particles in the 1d column. The required number of iterations grow roughly linear with the number of particles given a constant error tolerance.

## 2.3 The projected Gauss-Seidel method

The projected Gauss-Seidel (PGS) method is an extension of Gauss-Seidel used to solve MLCP's such as the one in equation 2.4. The PGS algorithm does this by treating each contact problem sequentially and then iterates until all contact laws are fulfilled to a specified error or number of iterations is reached.

Consider an example system of equations for NDEM:

$$\begin{bmatrix} \mathbf{M} & -\mathbf{G}^T \\ \mathbf{G} & \mathbf{\Sigma} \end{bmatrix} \begin{bmatrix} \mathbf{v} \\ \lambda \end{bmatrix} = \begin{bmatrix} \mathbf{p} \\ \mathbf{q} \end{bmatrix} \tag{2.14}$$

with the friction conditions that $\lambda_n \geq 0$, $|\lambda_t| \leq \mu|\lambda_n|$.

Let the submatrices $\mathbf{M}$ and $\mathbf{\Sigma}$ be block diagonal and $\mathbf{G}$ block sparse. The system can be split onto Schur complement form like:

$$(\mathbf{G}\mathbf{M}^{-\mathbf{1}}\mathbf{G}^{\mathbf{T}} + \mathbf{\Sigma})\lambda = \mathbf{q} - \mathbf{G}\mathbf{M}^{-\mathbf{1}}\mathbf{p} \tag{2.15}$$

$$\mathbf{v} = \mathbf{M}^{-\mathbf{1}}\mathbf{p} + \mathbf{M}^{-\mathbf{1}}\mathbf{G}^{\mathbf{T}}\lambda \tag{2.16}$$

The matrix $\mathbf{G}\mathbf{M}^{-\mathbf{1}}\mathbf{G}^{\mathbf{T}} + \mathbf{\Sigma}$ in equation 2.15 is called the schur matrix $\mathbf{S}$. This matrix can be split similar to the Gauss-Seidel split in equation 2.11 like: $\mathbf{S} = \mathbf{L} + \mathbf{D} + \mathbf{L}^{\mathbf{T}}$, $\mathbf{D}$ being block diagonal and $\mathbf{L}$ is strictly lower triangular. Equation 2.15 can be solved iteratively; denoting iterations by k and block indices with $\alpha$, $\beta$ it can be written in the form:

$$\mathbf{D}_{(\alpha\alpha)}\lambda_{\mathbf{k+1}}^{\alpha} + \sum_{\beta<\alpha} \mathbf{L}_{(\alpha\beta)}\lambda_{\mathbf{k}}^{(\beta)} = \mathbf{q}_{(\alpha)} - \mathbf{G}_{(\alpha)}\mathbf{M}^{-\mathbf{1}}\mathbf{p}_{(\alpha)} \tag{2.17}$$

with $\mathbf{D}_{(\alpha\alpha)} = \sum_a \mathbf{G}_{[a]}^{(\alpha)} \mathbf{M}_{[aa]}^{-1} \mathbf{G}_{[a]}^{(\alpha)T} + \sum_{(\alpha\alpha)}$. Adding and subtracting $\mathbf{D}_{(\alpha\alpha)}\lambda_k^\alpha$ to this the following update formula is obtained:

$$\mathbf{D}_{(\alpha\alpha)}\lambda_{k+1}^{(\alpha)} + r_k^{(\alpha)} - \mathbf{D}_{(\alpha\alpha)}\lambda_k^{(\alpha)} = 0 \tag{2.18}$$

The solution to this equation must satisfy the complementarity condition. The residual vector becomes:

$$\mathbf{r}_k^\alpha = \mathbf{S}_{(\alpha\alpha)}\lambda_k^{(\alpha)} + \mathbf{G}_{(\alpha)}\mathbf{M}^{-1}\mathbf{p}_{(\alpha)} - \mathbf{q}_{(\alpha)} = \mathbf{G}_{(\alpha)}\mathbf{v}' - \mathbf{q}_{(\alpha)} \tag{2.19}$$

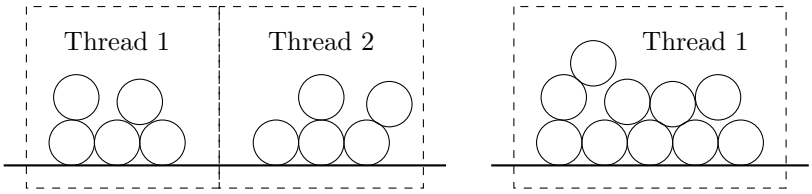
Where $\mathbf{v}' = \mathbf{M}^{-1}\mathbf{p} + \mathbf{M}^{-1}\mathbf{G}_{(\alpha)}^T\lambda_k$. In the projected Gauss-Seidel method, first the normal component is solved, then if $\lambda_{n,k+1}^\alpha > 0$, the tangential components are solved and projected onto the cone surface if it was outside.

$$\lambda_{t,k+1}^{(\alpha)} \leftarrow proj_{\mu\lambda_{n,k+1}^{(\alpha)}}\left(\lambda_{t,k+1}^{(\alpha)}\right) = min\left(\frac{\mu\lambda_{n,k+1}^\alpha}{|\lambda_{t,k+1}^{(\alpha)}|}, 1\right) \cdot \lambda_{t,k+1}^{(\alpha)} \tag{2.20}$$

This derivation from equation 2.14-2.20 is taken from the work by Servin et. al[12] (appendix B, equations 45-51). A full description of the projected Gauss-Seidel algorithm applied on NDEM can also be found in appendix B in the same reference.

## 2.4 Parallelization of projected Gauss-Seidel

One of the main difficulties when implementing a parallel version of PGS is the dependencies of the bodies in the system. To illustrate this consider figure 2.5 below.




(a) Trivially parallelized (b) Not trivially parallelized

Figure 2.5: Two different particle pile configurations

In figure 2.5(a) it is trivial to parallelize the pile configuration since the two piles found dont have any contact dependencies. The figure 2.5(b) however is not trivially parallelized since all the particles are found in the same contact network. Now imagine having 30K or 1 million particles in a pile and they all belong to the same contact network. In order to parallelize large connected contact systems one has to do it in a systematic way, the method used to parallelize the solver in AgX is described below.

The PPGS implementation divides 3D space in the simulation into cubic cells. There are three different kind of areas in the partitioning; internal, edge and corner. This is visualised in figure 2.6. The contacts in the particle system are then assigned to one of these three areas.

Dividing up the contact network in this way and solving the different zones in a systematic way solves the problem of the dependencies. One can now construct a graph that the solver can follow so it knows what zones can run in parallel and which zones has to wait for others to finish before they can start. One example of such a solve graph is shown in figure 2.7.

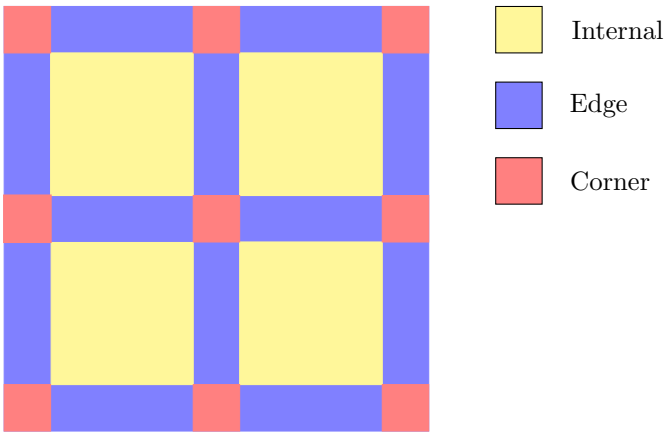



Figure 2.6: The types of zones used in the PPGS partitioning.

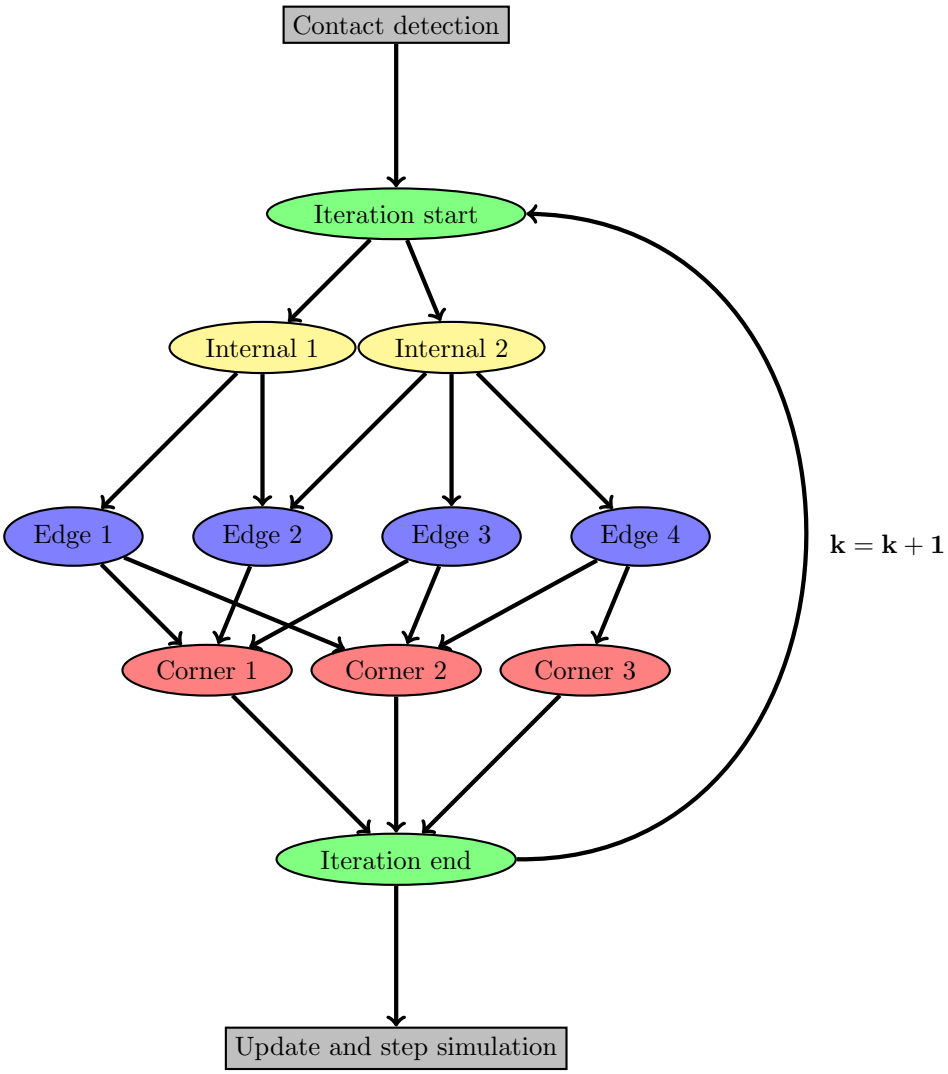



Figure 2.7: Zone dependencies in the PPGS implementation. **k** indicates the number of PGS iterations.

In this solve graph each node is a solve job. All the edges can execute in parallel and for example corner 1 can start before edge 4 is done since these have no direct

dependencies.

### 2.4.1 Speedup of the parallel projected Gauss-Seidel implementation

Some initial tests of the potential speedup of the PPGS implementation have been made at Algoryx[3]. One scene used was the one in figure 2.8 below. In this scene the contacts have been colored to their respective partitions that the PPGS algorithm put them in.

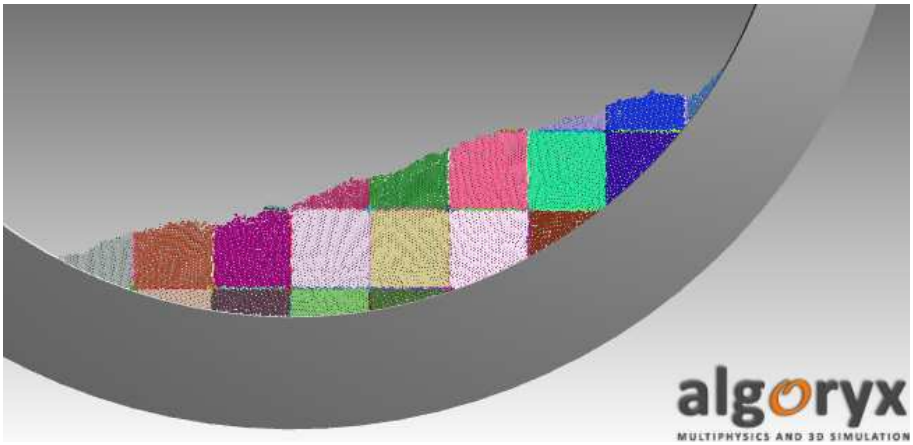



Figure 2.8: A rotating drum test scene with 100,000 particles used when measuring speedup of PPGS. The colours show different partitions. Image taken from reference[3].

In figure 2.9 from the same internal document at Algoryx[3], the achieved speedup is plotted against number of threads. The green line is speedup with jacobian caching and the blue line is without. If Jacobian caching is not used it means that the Jacobian is calculated again in every Gauss-Seidel iteration when the data is in the CPU. The speedup in the graph only measures the speedup of the Gauss-Seidel iterations, not including overhead or the other parts of the simulation loop such as collision detection etc. One can see that the maximum speedup when not caching the Jacobian for this case is around 6 for 8 threads and then it saturates.

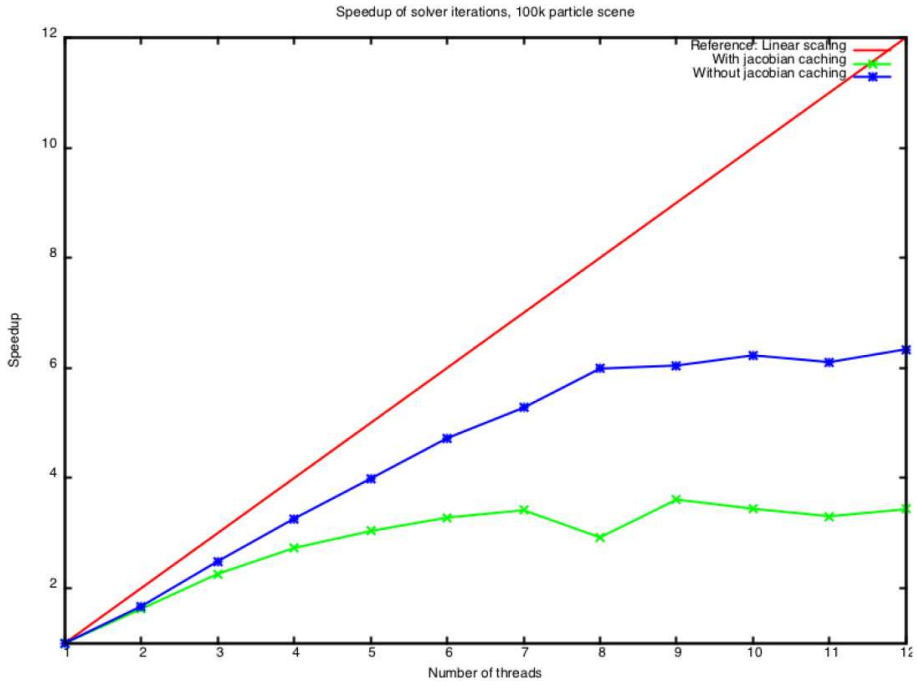



Figure 2.9: Speedup of PPGS vs number of threads used. Plot is taken from reference[3].

## 2.5 Multigrid projected Gauss-Seidel

The theory and numerical methods for multigrid is not well developed for contacting multibody systems. This theory section will try to explain in a simple way what is meant with multigrid in the context of this thesis and applying it to a DEM simulation.

Standard multigrid has been around for some time and the main idea of it is to speed up the convergence of the iterative numerical method. It does this by solving a coarse problem, similar to interpolation between a coarse and fine grid, see the book Multigrid[13] for a more in-depth description. This book however only covers multigrid for FEM and FDM family of methods and not much can be found for DEM multigrid.

The main idea for a general multigrid 3d particle system for DEM is as follows. Let 3d space be divided into partitions. The particles in these partitions is then merged into rigid bodies. These merged rigid bodies are solved and contact forces are obtained. The contact forces are then used to solve the internal systems of the merged rigid bodies.

For simplicity we first consider a quasistatic 1d column. The theory is described in chapter 2.5.1. A prototype multigrid DEM implementation has been developed for AgX and results of this can be found in section 5.3.

### 2.5.1 A first prototype multigrid solver for 1D PGS

Say we have a 1d column of 12 particles, see figure 2.10, we then divide this into three partitions each containing 4 particles. A merged rigid body is denoted by $\bar{r}_i$, the bar meaning it is a merged macro body. A particle within a macro body is denoted $r_i^j$, i being the macro body and j being the internal particle. Geometries that are in contact with the particles are called $gi$, i being index. In the following example we have only ground as a external geometry so it is denoted $g0$. Contacts are denoted with a capital C.

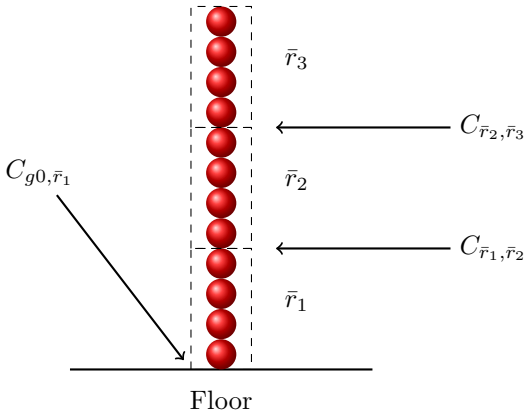



Figure 2.10: Conceptual drawing of the partitioning

From this point we sum up all the individual particles in the partitions into rigid bodies, so for example the center of mass of $\bar{r}_1$, $\bar{r}_2$ would be computed using equation 2.23, visualized by figure 2.11.

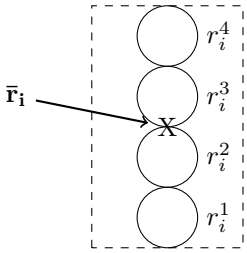



Figure 2.11: Calculating the metrics of a rigid body from its internal particles

The force acting on the rigid body is aquired by adding the forces of the individual particles

$$\mathbf{F}_{\mathbf{r_i}} = \sum_{k=1}^{N} \mathbf{F}_{\mathbf{r_i^k}} \tag{2.21}$$

The total mass of $\bar{r}_i$

$$M_{\bar{r}_i} = \sum_{k=1}^{N} m_{r_i^k} \tag{2.22}$$

The center of mass of $\bar{r}_i$ can be calculated

$$\bar{\mathbf{r}}_\mathbf{i} = \frac{1}{M_{\bar{r}_i}} \sum_{k=1}^{N} m_{r_i^k} \mathbf{r_i^k}, \tag{2.23}$$

where $\mathbf{r_i^k}$ is the particles position from origin. The inertia tensor and some other properties can also be calculated from the particles properties but is not shown here.

First solve the macro rigid body system using a direct solver, consider figure 2.12.

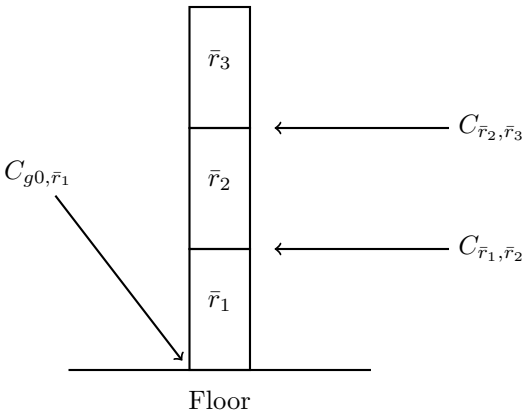



Figure 2.12: Solving the macro system

From this one aquires exact solutions to the contacts $C_{g0,\bar{r}_1}$, $C_{\bar{r}_1,\bar{r}_2}$, and $C_{\bar{r}_2,\bar{r}_3}$. Velocities and positions from this solution are discarded and not updated. Now let each partition of the system run as a separate subsystem in parallel with contact information from the macro-solve, visualized in figure 2.13.

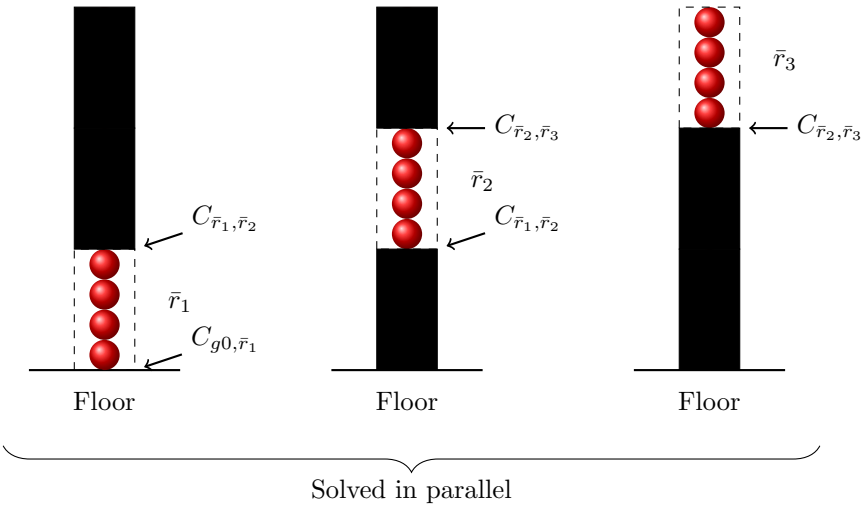



Figure 2.13: Solving of micro-states in parallel

Each micro-solve in parallel can be solved using an iterative solver, for example projected Gauss-Seidel.

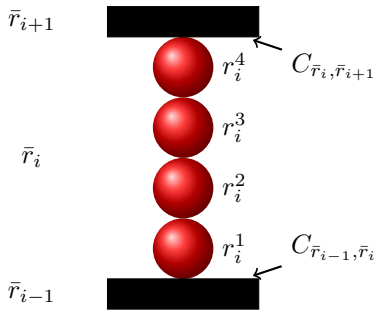



Figure 2.14: Solving a micro-state

The contact force of $C_{\bar{r}_{i-1},\bar{r}_i}$ from the macro solve is applied to particle $r_i^1$ and the

contact force of $C_{\bar{r}_i,\bar{r}_{i+1}}$ is applied to particle $r_i^4$ in the micro-solve, shown in figure 2.14. These contact forces will be constant for each iteration within a timestep. This micro system is run for $N_{it}$ number of iterations, until a desired accuracy is reached. After all the micro-solves has finished in parallel the algorithm applies the forces and steps the system.

**Algorithm 1** Multigrid prototype solver for DEM

```
for each timestep do
    Partition the particles into zones of rigid bodies called rb_i
    (*) Compute the properties of the rigid bodies rb_i in each zone from its internal particles
    (X) Solve the macro rigid body system with a direct solver
    for each r̄_i do
        for N_it do
            (*) Solve micro-system r_i using contact forces from contacts C_{r̄_{i-1},r̄_i} and C_{r̄_i,r̄_{i+1}}
        end for
    end for
end for
```

Note: (*) can be run in Parallel and (X) has to be run in seriel.

# Chapter 3

# Tools and implementation

## 3.1 AgX framework

The AgX framework[1] is a modular physics simulation toolkit written in C++. It is a toolbox which contains all the tools necessary to perform a number of different types of simulations from stable and robust real-time simulations of mechanical systems to the simulation of millions of granules, not yet in real-time. One of the main purposes of the framework is to provide accurate simulations for the engineering and scientific community and to be used in real-time training simulators.

The framework contains a number of different numerical solvers, both machine-precision direct solvers and iterative solvers for fast approximate solutions. The simulation scene is modeled either directly through C++ or via the scripting language Lua[8]. Algoryx has implemented Lua scripting into AgX and the lua files used for modeling scenes are conveniently named .agxLua.

AgX is built with parallelism in mind from the start and the collision detection is currently fully parallelized. The tools for extracting large amounts of data from the simulations is supported through a journal system. This was used extensively in this thesis to collect simulation data and save it to disk. Efficient data structures to hold information about rigid bodies, particle systems etc exists in AgX. Parameters for contact materials such as Young's modulus, Poisson's ratio etc can also be set.

## 3.2 Analysis pipeline

A pipeline was developed to handle the simulation and post-processing of the different scenes. This pipeline was written from scratch in the scripting language Python[9] which suits this purpose well. Python is open-source and with a couple of packages such as numpy, scipy and matplotlib it is very powerful when doing scientific computing and plotting. The format HDF5[5] was chosen to store the data for the simulation as it is an open format, supported by many different languages such as Matlab, Octave, Python etc, and AgX had built-in support for it.

### 3.2.1 Simulation pipeline

The simulation pipeline has a main python and agxLua script file for each scene, which completely describes one simulation run. The simulation parameters can easily be changed in the python file, such as number of Gauss-Seidel iterations, how many threads

to run, whether or not to use PPGS and so on. A graph visualization of the pipeline can be see in figure 3.1 below.

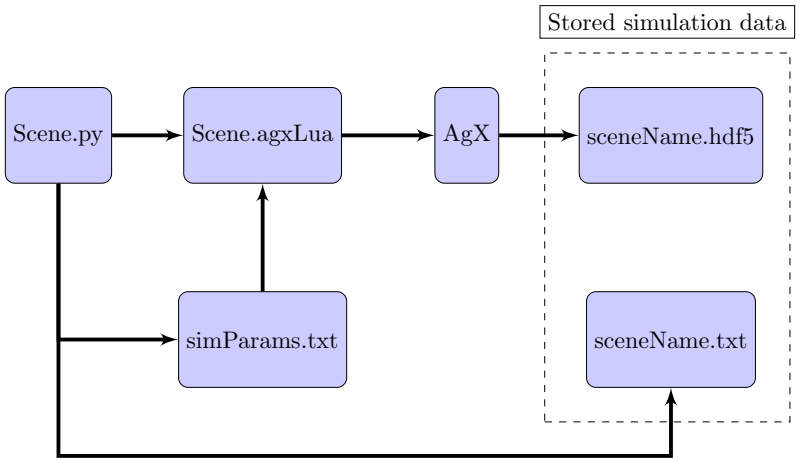



Figure 3.1: Visualization of the simulation pipeline

The file sceneName.txt can hold any custom information important to the simulation, such as the parameters in table 4.1.

### 3.2.2 Post-processing pipeline

The post-processing pipeline reads data from the hdf5 file and txt file created by the simulation pipeline. These two pipelines are completely separated and this design choice is deliberate in order to maximize compatibility with other simulations as well as future releases.

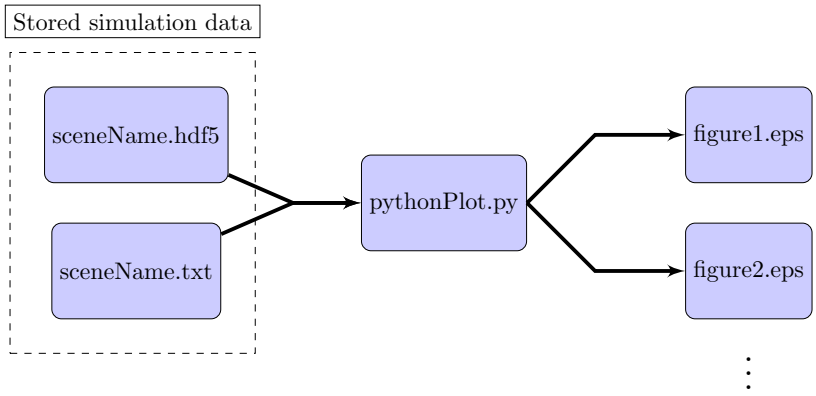



Figure 3.2: Visualization of the postprocessing pipeline

## 3.3 AgX multigrid prototype implementation

The multigrid prototype algorithm in the theory section, algorithm 1, was implemented in AgX on a hard-coded, 1d column. This was done by creating sub-simulations in the main simulation loop that were responsible for solving the merged bodies with a direct solver and the micro-solves using PGS. A 3D version of the multigrid algorithm was also implemented but some difficulties arose which are discussed in chapter 6.2.1.

# Chapter 4

# Simulation

The simulation part of the thesis is described in detail below, both for the scenes used to compare the serial and parallel solver as well as the scene used for testing the multigrid prototype.

## 4.1 Test systems

This section will describe the test system setup for the PPGS comparison as well as the multigrid system setup.

### 4.1.1 Test scenes for comparing the parallel and serial solver

Three test scenes were implemented to compare the serial and parallel Gauss-Seidel implementation. A 1D column of 100 particles, seen in figure 4.1, a 3D Cylinder seen in figure 4.2 and a rotating drum seen in figure 4.3. All the scenes had the same simulation parameters, seen in table 4.1. The 3D cylinder have two different sizes, one with diameter 9D and one with diameter 15D, where D is particle diameter so (this specific case 0.09 m and 0.15 m). They are both tall enough to hold as many particles as was needed for the simulation, 2.1 m. The drum has an inner diameter of 80D with 64 lifters as described in figure 4.4. This rotating drum also comes in two versions, one with depth 10D and one with depth 30D. The rotating speed of the drum was set to 0.63 rad/s in the direction of the lifters as seen in figure 4.4.

The drum was created in the CAD software SpaceClaim [4] as a so-called triangular mesh. This triangular mesh was imported into AgX and set in motion.

Table 4.1: Simulation parameters for the test scenes, h is timestep

| | |
|---|---|
| Particle Diameter | 0.01 [m] |
| Particle density | 2500 [kg/$m^3$] |
| Young's modulus | 5 $\cdot 10^6$ [Pa] |
| Poissons ratio | 0.3 |
| Restitution coefficient | 0 |
| Time step | 0.01 [s] |
| Damping time | $2 \cdot$ h = 0.02 [s] |

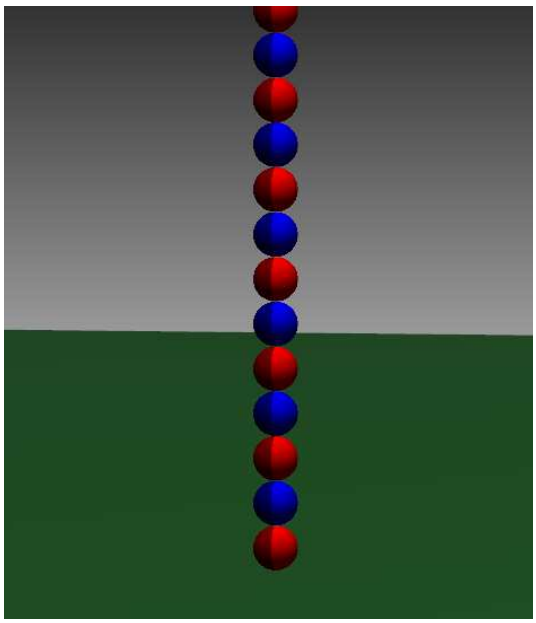


Figure 4.1: A 1d column test scene with 100 particles.

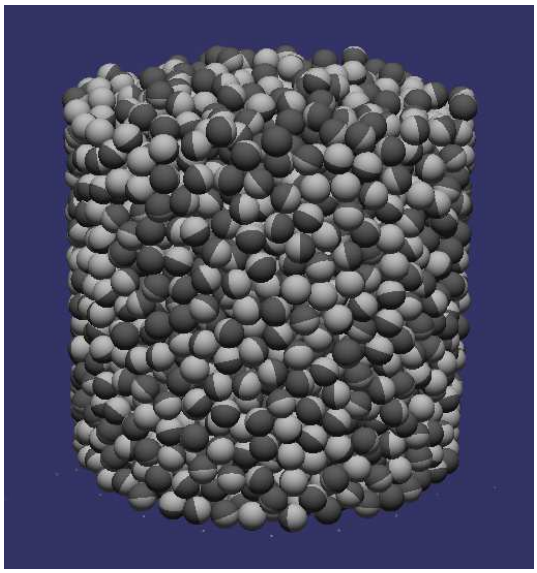


Figure 4.2: A 3d cylinder test scene containing 30000 particles with an inner diameter of 15D.

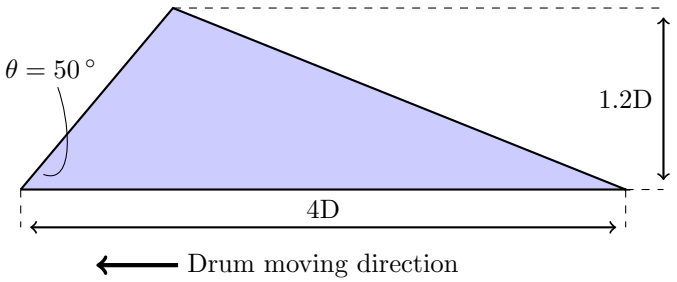



Figure 4.4: Design of the drum lifters, D is particle diameter. 64 of these are placed symmetrically in the drum.

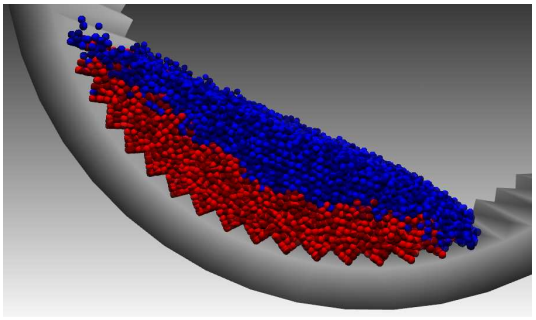


Figure 4.3: A rotating drum test scene containing 7500 particles with a depth of 10D.

### 4.1.2 Test scene for studying multigrid

A visualization of the 1d multigrid test scene can be seen in figure 4.5. Starting from the left, column 1 is solved with a direct solver, column 2 is solved with an iterative Gauss-Seidel solver and column 3 is the multigrid column, where a red sphere indicates the bottom of one partition and a blue sphere indicates the top of a partition. The same parameters are used as in table 4.1 except for Young's modulus which was chosen to be $5{\cdot}10^{11}$.

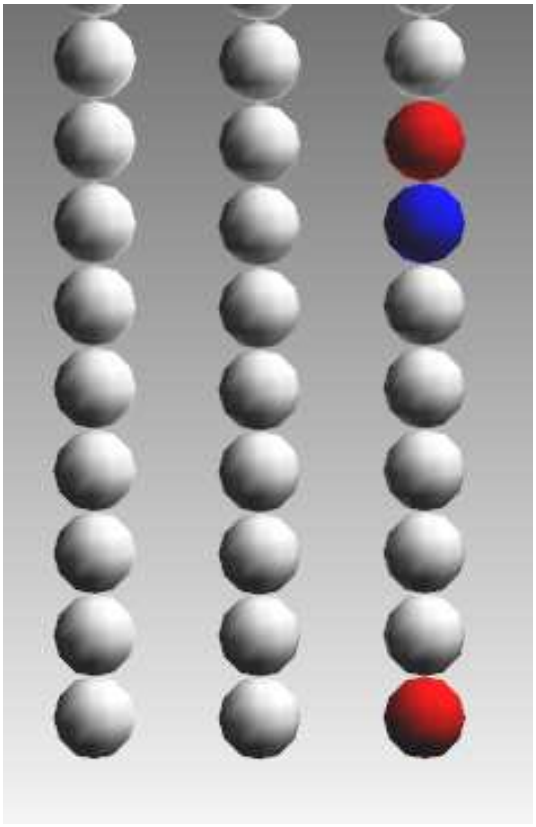


Figure 4.5: A 1d column multigrid test scene where the left column is solved with a direct solver, middle column with an iterative PGS solver and the right column is solved using multigrid.

## 4.2 Measurements and metrics

A lot of data was extracted from the simulations in order to do efficient and accurate post-processing. The graph in figure 4.6 shows what data was extracted from the simulation and what kind of plots was produced from these.

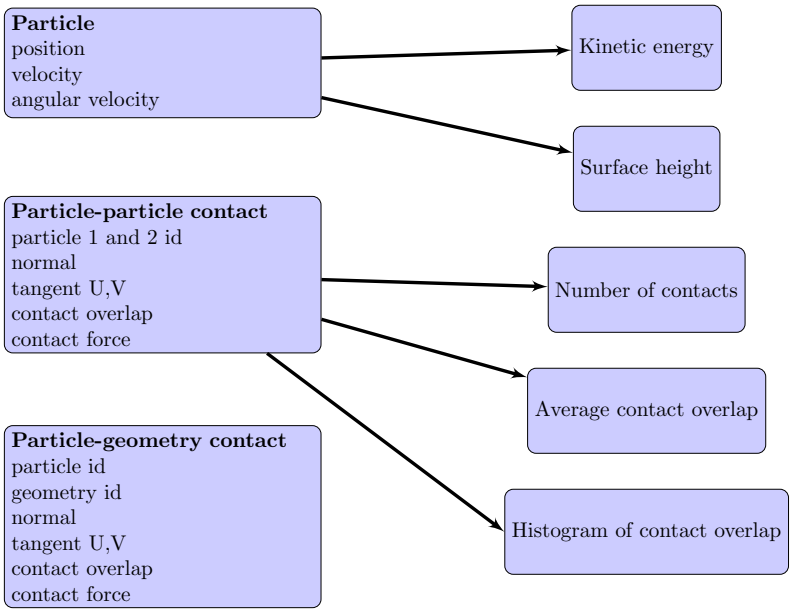



Figure 4.6: A visualization of what was measured and in the simulation (left) and which plots were produced from this (right).

## 4.3 Test procedure

All the test scenes were run with the parameters in table 4.1. The scenes were run for 10-20 seconds depending on the size of the systems. The particles in the 1D column scene was created with a distance of 2D from each other and so the initialization of this scene is exactly the same for every run. This is not the case in the 3d cylinder and rotating drum scene since these use a particle emitter to initialize the particles. What a particle emitter does is it creates particles at a given rate in random positions inside the emitter. For the 3d cylinder the emitter was moving with a constant speed along the z-axis and in the rotating drum scene the emitter was placed in a fixed position inside the drum, which can be seen as the small cylinder in figure **??**.

Using the particle emitter to initialize the 3d cylinder and rotating drum scenes was a good way to smoothly initialize the simulation as it can sometimes be problematic to initialize it in other ways. A side effect of using the particle emitter is that two consecutive simulation runs does not behave exactly the same as the particles are spawned in random positions. This is not a major problem though if we average the metrics over time, for example looking at the average kinetic energy over hundreds of time steps.

The data from the simulations were recorded using a Journal system that existed in AgX. The simulations were run at 100 Hz and any evenly divisible recording frequency could be chosen to dump data to disk. Every 10th timestep was chosen for data recording for a tradeoff between accurate data measurement and disk space needed.

# Chapter 5

# Results

The results for the simulations are presented below as well as an explanation of the metrics and symbols used to describe what the data in the plots are.

## 5.1 Notations used in the plots

The data in most of the plots have been normalized against some value. While the horizontal axis of all the plots is time in seconds, the vertical axis could need some explaining. The normalized contact overlap is denoted $\epsilon_c$, defined as $\epsilon_c = g/D$ where D is the particle diameter and g is overlap. Kinetic energy $E_{kin}$ is normalized by dividing it with the potential energy to lift all particles in the simulation one particle diameter, $E_{mgd} = N_p m_p g D$, where $N_p$ is the number of particles, $m_p$ is particle mass, $g = 9.82$ and $D$ particle diameter. We denote the normalized kinetic energy by $E_{norm}$ so that $E_{norm} = \frac{E_{kin}}{E_{mgd}}$. Number of contacts is denoted $N_c$. The surface height in figure 5.11 is calculated by inspecting the 20 particles in the simulation that are at the highest height. The value for the surface height is then the average of the height of these 20 particles.

In all of the plots an average of the whole time series is represented with the symbol $\langle * \rangle$ where * is the data the average is calculated on. If there is a bar over a metric, for example contact overlap, like so $\bar{\epsilon}_c$, this means that the average of all the particles contact overlap for this time step is taken. The contact overlap, $\epsilon_c$, is normalized by dividing it with the particle radius, $R$.

## 5.2 Comparison of parallel and seriel projected Gauss-Seidel

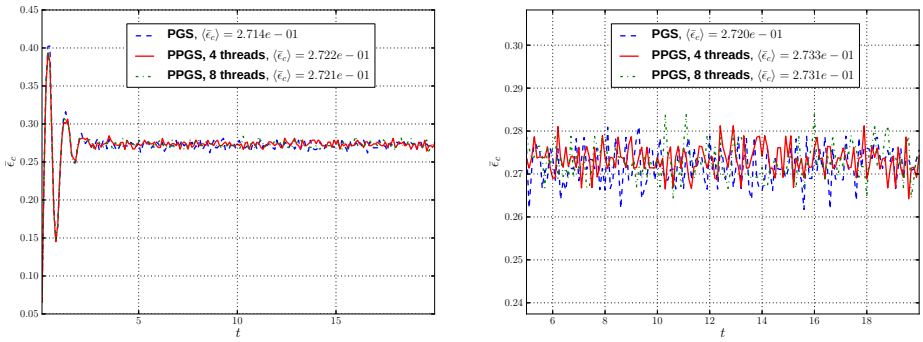
The three test scenes described in chapter 4 were run with serial projected Gauss-Seidel, 4 threads PPGS and 8 threads PPGS, and these three configurations in 100 and 500 iterations of gauss-Seidel.

A percentage can be calculated on the error of how much the PGS time series and the PPGS time series differ from eachother. This error calculation should only be used when a steady state has been reached in the simulation and there seems to be no drifting of the measurements. Let $K$ denote one of the time series in the plots. The error is calculated in the following way:

$$error = \frac{|\langle \bar{K}_{PGS} \rangle - \langle \bar{K}_{PPGS} \rangle|}{max(K_{PGS}, K_{PPGS}) - min(K_{PGS}, K_{PPGS})} \tag{5.1}$$

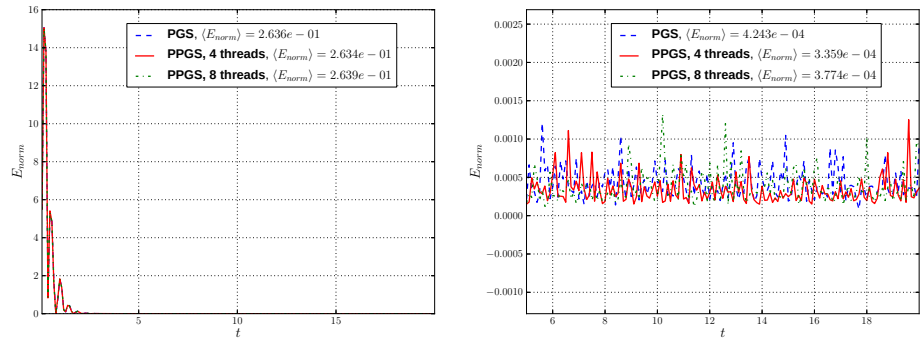
### 5.2.1 1D column

The 1D column is 100 particles tall with particles being placed a distance of 2D from each other. In the figures 5.1(a), 5.2(a) and 5.3(a) the whole simulation run can be seen from 0 to 20 seconds. Looking at the contact overlap (5.1(a)) and kinetic energy (5.2(a)), the simulations seems to go from an initialization state to a steady state at and after 4 seconds. The figures 5.1(b), 5.2(b) and 5.3(b) are from time 5 to 20 seconds and shows the simulations when they are in their steady state. The interesting numbers to look at for the comparison are in the (b) plots, the average values $\langle * \rangle$ which should give an indication of how much they differ.



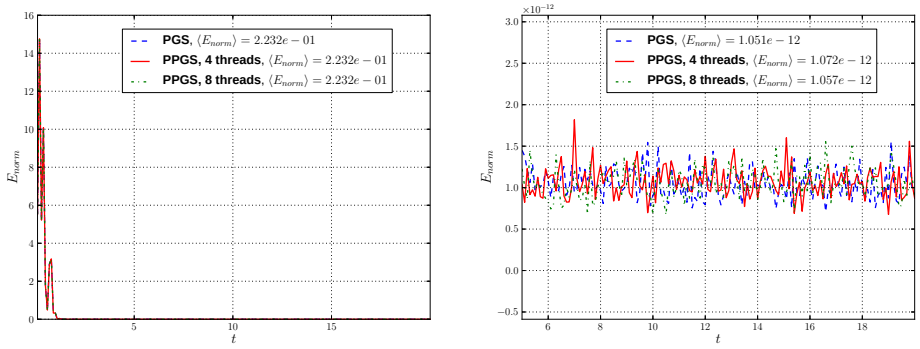

(a) 0-20 seconds (b) 5-20 seconds

Figure 5.1: Average contact overlap for a 1d column with 100 particles simulated using 100 iterations.




(a) 0-20 seconds (b) 5-20 seconds

Figure 5.2: Normalized kinetic energy for a 1d column with 100 particles simulated using 100 iterations.




(a) 0-20 seconds

(b) 5-20 seconds

Figure 5.3: Normalized kinetic energy for a 1d column with 100 particles simulated using 500 iterations.

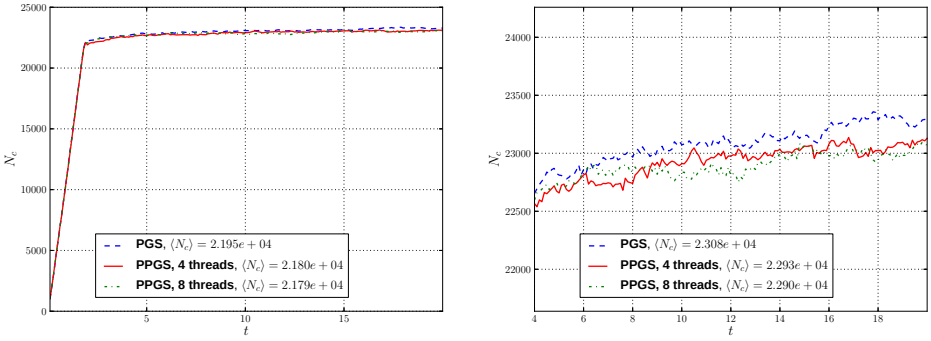
The simulations for the 1D column are in the time period 5-20 seconds in a steady state. The error calculation formula in equation 5.1 was used to approximate the error, the results is collected in table 5.1 below for the presented plots.

Table 5.1: Quantitative error in the 1D column test.

| Measurement | $N_{it}$ | Error | Figure |
|---|---|---|---|
| Contact overlap | 100 | 6.5% | 5.1(b) |
| Normalized kinetic energy | 100 | 9.8% | 5.2(b) |
| Normalized kinetic energy | 500 | 2% | 5.3(b) |

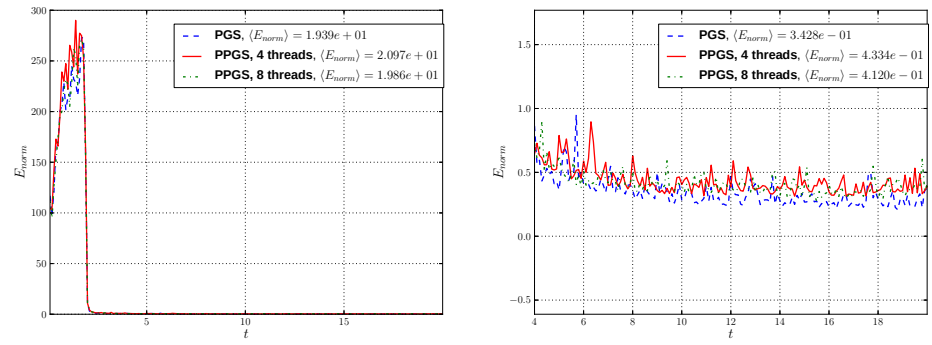
### 5.2.2 3D cylinder, inner diameter 9D

In the plots for the 3D cylinder the diameter of the cylinder is 9D and 7500 particles were used. In figure 5.4(a) and 5.5(a) there is an initialization phase from 0 seconds to around 2.5 seconds when the particles are being created by the particle emitter. The contact overlap in figure 5.6 is steadily decreasing over time. One can suspect that neither the 100 iteration simulation nor the 500 iteration simulation has reached a steady state after 20 seconds since the measurements are still drifting, but this is of little importance for the comparison of the solvers. The important measurement for comparison is the mean value in figures 5.4(b), 5.5(b) and 5.6(b) as well as the shape and $\mu$, *median*, $\sigma$ for the histogram in figure 5.7.



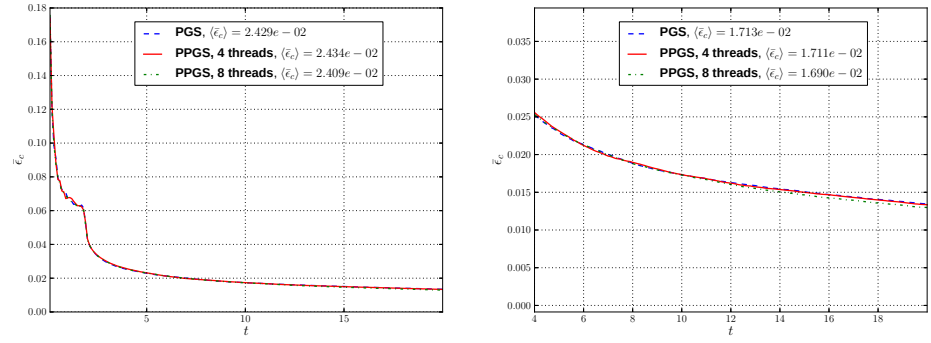

(a) 0-10 seconds (b) 4-20 seconds

Figure 5.4: Number of contacts for a 3d cylinder with an inner diameter of 9 particle diameters. The cylinder contains 7500 particles and is simulated using 100 iterations.



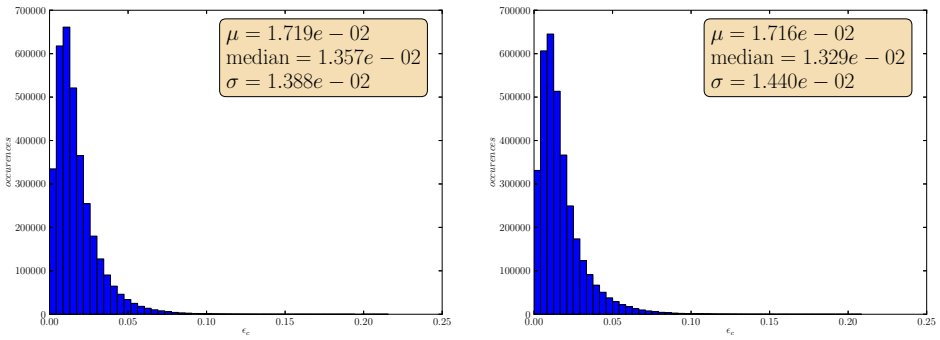

(a) 0-10 seconds (b) 4-20 seconds

Figure 5.5: Normalized kinetic energy for a 3d cylinder with an inner diameter of 9 particle diameters. The cylinder contains 7500 particles and is simulated using 100 iterations.




(a) 0-20 seconds (b) 4-20 seconds

Figure 5.6: Average contact overlap for a 3d cylinder with an inner diameter of 9 particle diameters. The cylinder contains 7500 particles and is simulated using 500 iterations.




(a) 4-20 seconds, serial

(b) 4-20 seconds, PPGS 4 threads

Figure 5.7: Histogram of the sum of the contact overlap. A 3d cylinder is used containing 7500 particles with an inner diameter of 9 particle diameters, simulated using 500 iterations.

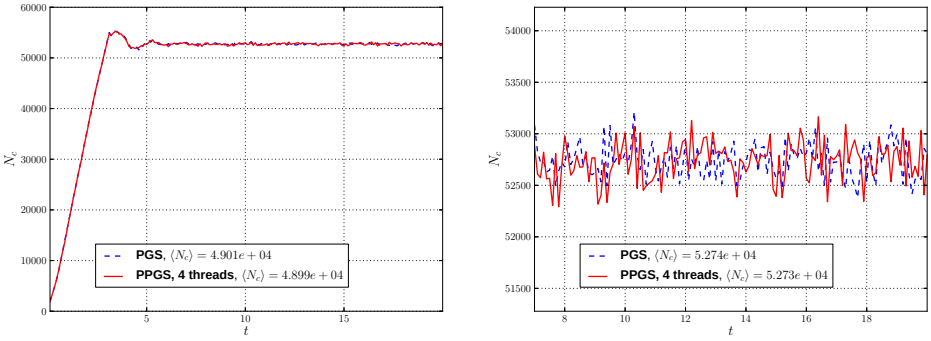
The error calculation in equation 5.1 is decent for steady state but not so good for the measurements we have here, when the values are still drifting. Instead we calculate the difference in percent of the mean values of PPGS and PGS, the worst case is taken when deciding between PPGS 4 thread and PPGS 8 thread.

Table 5.2: Quantitative error in the 3D cylinder test.

| Measurement | $N_{it}$ | Mean error | Figure |
|---|---|---|---|
| Number of contacts | 100 | 0.8% | 5.4(b) |
| Contact overlap | 500 | 1.3% | 5.6(b) |
| Histogram contact overlap | 500 | 2% | 5.7(b) |

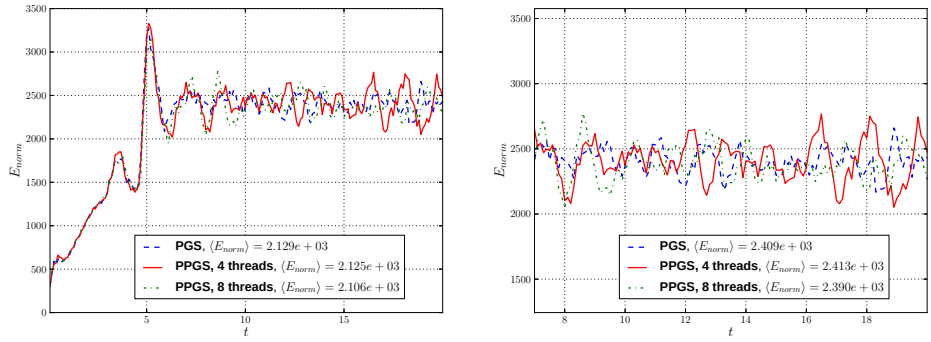
### 5.2.3 Rotating drum, depth 10D and 30D

Considering figures 5.8, 5.9, 5.10, 5.11 and 5.12 five different simulation configurations are shown for the rotating drum case. In figures 5.8(a)-5.11(a) the simulations seem to have an initialization state from 0 to 7 seconds, the data is not drifting after seven seconds. When the simulation reached 7 seconds it was considered to be in a steady state so this is what the figures 5.8(a)-5.11(b) depicts as well as the histogram comparison in figure 5.12.



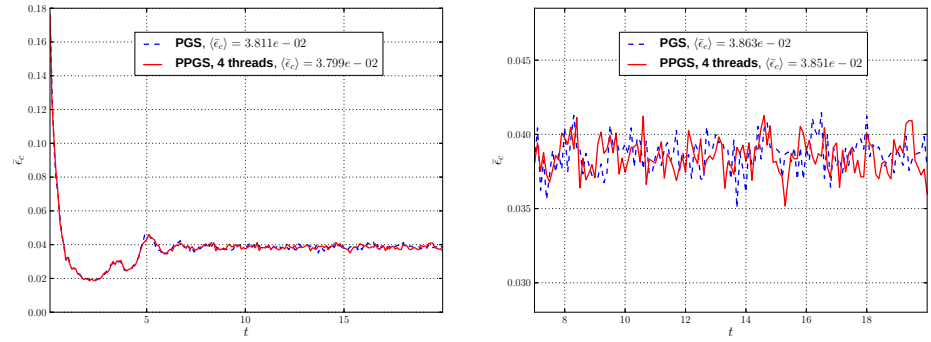

(a) 0-20 seconds (b) 7-20 seconds

Figure 5.8: Number of contacts for a rotating drum (depth is 30D) containing 25000 particles having a depth of 30 particle diameters. The simulation is run at 500 iterations.



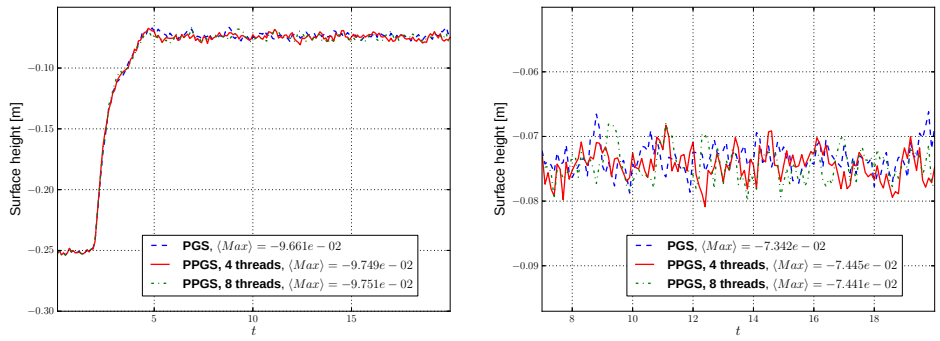

(a) 0-20 seconds (b) 7-20 seconds

Figure 5.9: Normalized kinetic energy for a rotating drum (depth is 10D) containing 7500 particles. The simulation is run at 100 iterations.



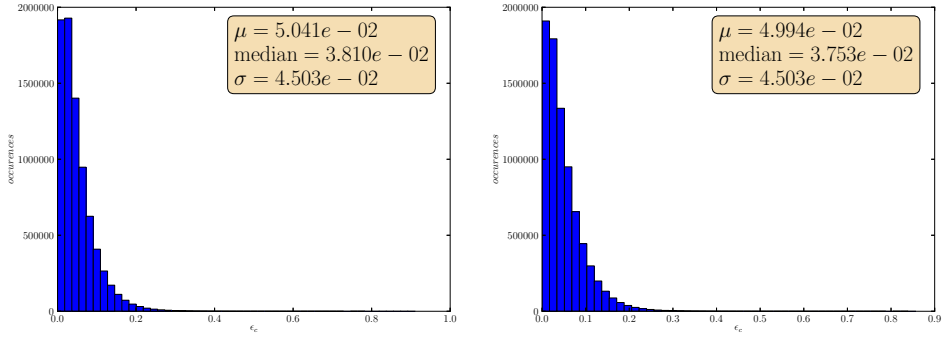

(a) 0-20 seconds (b) 7-20 seconds

Figure 5.10: Average contact overlap for a rotating drum (depth is 10D) containing 7500 particles. The simulation is run at 500 iterations.




(a) 0-20 seconds (b) 7-20 seconds

Figure 5.11: Surface height for a rotating drum (depth is 10D) containing 7500 particles. The simulation is run at 100 iterations.




(a) 7-20 seconds, serial (b) 7-20 seconds, PPGS 4 threads

Figure 5.12: Histogram of the sum of the contact overlap. A rotating drum (depth is 30D) is used containing 25000 particles. The simulation is run at 100 iterations.

The error formula in equation 5.1 is used to calculate the error in table 5.3. There is some variations in the data and after 7 seconds a steady state seems to have been reached.

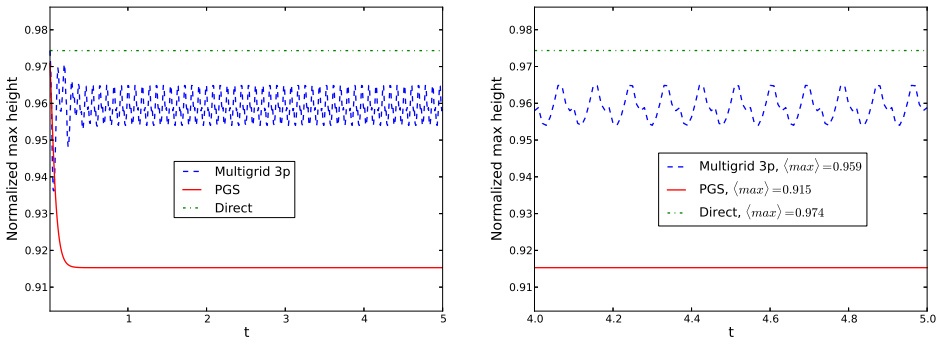
Table 5.3: Quantitative error in the rotating drum test.

| Measurement | $N_{it}$ | Error | Figure |
|---|---|---|---|
| Number of contacts | 500 | 1% | 5.8(b) |
| Normalized kinetic energy | 100 | 3.2% | 5.9(b) |
| Contact overlap | 500 | 1.8% | 5.10(b) |
| Surface height | 100 | 7.3% | 5.11(b) |

## 5.3 Comparison of the multigrid, direct and iterative PGS solvers.

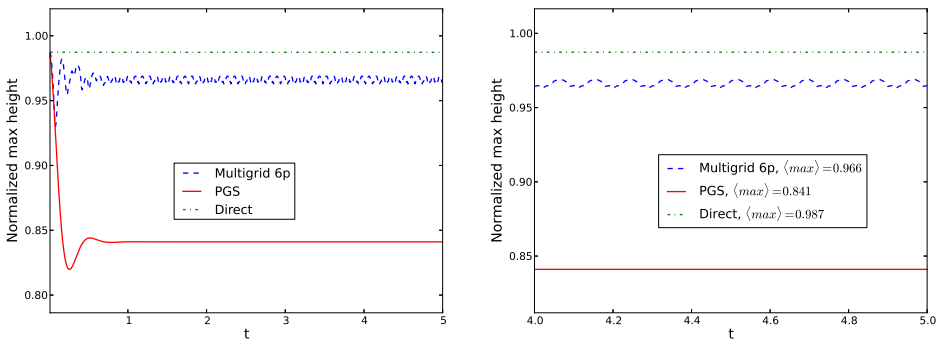
All the multigrid tests were done with 100 gauss-seidel iterations for the iterative column and for the micro-solves in the multigrid. The simulation parameters were chosen to be the same as the parameters in table 4.1 with the exception of Young's

modulus, which was set to $5 \cdot 10^{11}$ [Pa]. The height in figures 5.13, 5.14 and 5.15 is normalized so that 1.0 means no compression of the column in the simulation. Multigrid 3p means there are three partitions in the multigrid simulation, similar for multigrid 6p with six partitions.



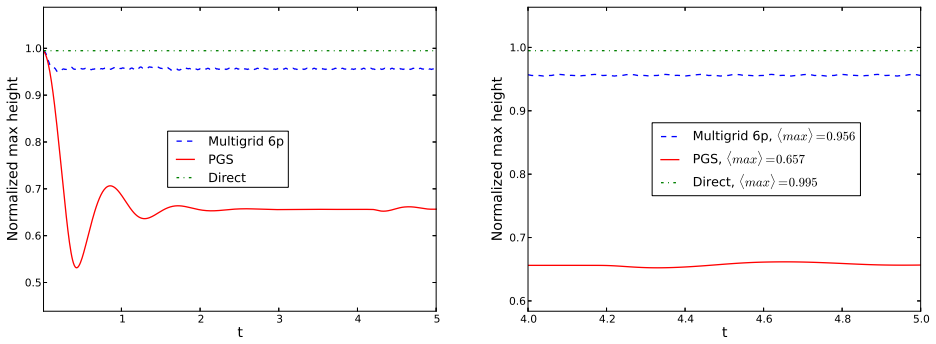

(a) 0-5 seconds (b) 4-5 seconds

Figure 5.13: Normalized maximum height of a 1d column with 20 particles solved using a direct, iterative and multigrid solver. The multigrid column is partitioned using three partitions.



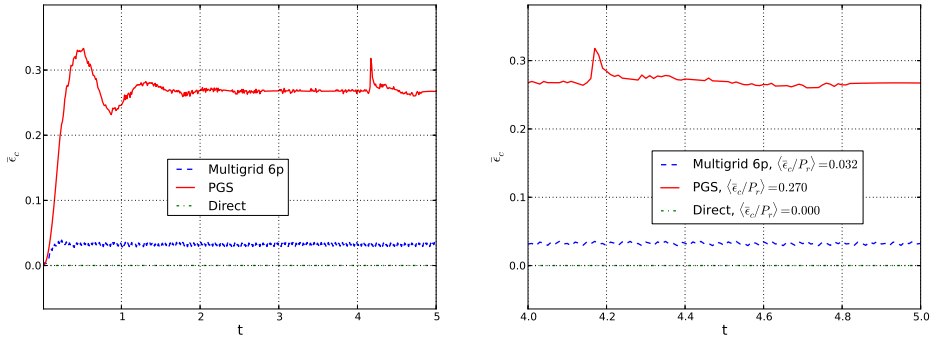

(a) 0-5 seconds (b) 4-5 seconds

Figure 5.14: Normalized maximum height of a 1d column with 40 particles solved using a direct, iterative and multigrid solver. The multigrid column is partitioned using six partitions.



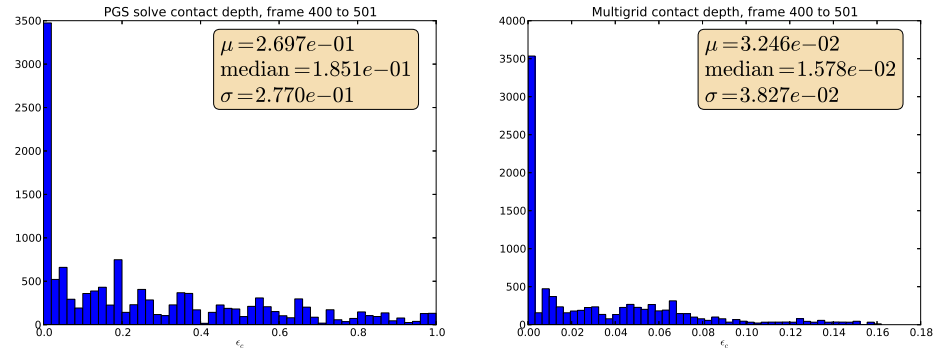

(a) 0-5 seconds (b) 4-5 seconds

Figure 5.15: Normalized maximum height of a 1d column with 100 particles solved using a direct, iterative and multigrid solver. The multigrid column is partitioned using six partitions.




(a) 0-5 seconds (b) 4-5 seconds

Figure 5.16: Average contact overlap of a 1d column with 100 particles solved using a direct, iterative and multigrid solver. The multigrid column is partitioned using six partitions.




(a) PGS, 4-5 seconds (b) Multigrid 6p, 4-5 seconds

Figure 5.17: Contact overlap histogram comparison for a 1d column with 100 particles solved with an iterative PGS solver and a multigrid solver using six partitions.

Table 5.4: The relative change in height between direct and multigrid / iterative

| Test system configuration | Relative height | |
|---|---|---|
| | MG/Direct | Iterative/Direct |
| 20 particles, 3 partitions | 0.984 | 0.937 |
| 40 particles, 6 partitions | 0.980 | 0.851 |
| 100 particles, 6 partitions | 0.96 | 0.66 |

In table 5.4 the relative height changes of the iterative and multigrid methods were compared for three different configurations. With the same iterations in multigrid and PGS, the multigrid method scales better with added number of particles in the 1d column. For the histogram comparison in figure 5.17 we can observe that the shape is similar but there is roughly a power of 10 difference in the median values. Another noticable thing is that the PGS histogram has some contact overlap at 1.0, indicating the particles have nearly passed each other.

# Chapter 6

# Discussion and Conclusions

In this chapter the results of the comparison between PGS and PPGS will be discussed as well as the results of the multigrid prototype implementation. First the results will bne discussed and after this the conclusions will be stated with bullet points.

## 6.1 The parallel projected Gauss-Seidel solver

Looking at all the figures from 5.1 to 5.12 for the three test scenes, the PGS and PPGS plots follow each other closely. The steady state in 1D column in figures 5.1(b)-5.3(b) we notice fluctuations but the mean values only differ 0.5% for the contact overlap in figure 5.1(b). Similar in the rotating drum case, figure 5.8-5.11, the behaviour of the data in the plots follow each other closely. The shape of the histograms in figures 5.7 and 5.12 are very similar and their quantitative values close (1.5% error in the mean value of the rotating drum histogram).

Some randomness will be introduced into the 3D cylinder and rotating drum simulations by means of the particle emitter. One other thing contributing to the difference between the PGS and PPGS simulations is the non-deterministic way that PPGS behaves, described in the background chapter 1.1.

- My conclusion is that the physics of the parallel and serial solver are behaving very similar, considering contact overlap there is a 1.3% (table 5.2) error in the 3D cylinder case and a 1.8% (table 5.3) error in the rotating drum case.

## 6.2 The multigrid prototype

In the plots in results section 5.3 the direct solution is considered exact and we wish to come as close as possible to this solution using the multigrid method given the same number of iterations as the PGS method. Considering the 100 particle column in figure 5.15 we can see that the particles in the direct solver is barely moving (compressed 0.5%) and the particles in the iterative solver is compressed by 34.3% while the multigrid column is compressed by 4.4%. The oscillation of the 1d multigrid method could break the strong chains in the force network, but was not examined further since this was a small test case for a 1d column. The force chains must not be altered when solving the system using the multigrid method as this will change the physics of the granular matter. The oscillation could be due to the fact that the partitions are hard-coded in the test implementation, that is for example particle 1-6 would always merge to a body, regardless of whether these particles were in contact or not.

- Given the same number of iterations, the 1d multigrid implementation behaves closer to a direct solver than the PGS solver. For the 100 particle case there is a compression of 4.4% for multigrid versus 34% for the PGS method compared to a direct solver (table 5.4).
- The multigrid method scales better than the PGS method when the number of particles are increased in the column (table 5.4).

### 6.2.1 Attempt at a 3D multigrid solver

An attempt was made to implement algorithm 1 for the 3D case. A 3D cylinder was used with 300 particles in it divided into 3 partitions with roughly 100 particles in each. In some test runs around 70 contacts were found between partition 1 (bottom-most partition) and its cylinder geometry and partition 2. When these particles were merged into one rigid body and solved using a direct solver, around 10 contacts were given non-zero forces and the rest of the contact forces were zero. This was at first unexpected but after some discussions it is exactly how the solver behaves, the problem is strongly overdetermined. One physical behaviour that should be preserved is the weak and strong force chains in the force network. If this problem is not carefully handled then multigrid in 3D will break the force chains and a lot of the characteristic physics of granular material is broken. One idea of how to further develop multigrid in 3D is briefly discussed here.

One possible solution to this problem is to first sum up all the contact forces. These contact forces should then be distributed among the 70 contacts found in the collision detection stage of solving the merged body. The contact overlap in the contacts can be used as a measure of how big the force should be at this contact, so distribute the total force amongst the contacts according to the size of the contact overlap.

This is not difficult to test but not enough time was left in the project.

## 6.3 Further work

I consider the comparisons I made in this thesis between PGS and PPGS to be sufficient. I can not find any suspect behaviour that should need further analysis for the behaviour of the PPGS solver.

The development of the 3D multigrid is interesting and should be explored further. As explained in the conclusion chapter 6.2.1 there are some ideas on how to take this prototype implementation to the next level. The simulation and post-processing pipeline developed for this thesis to compare different simulation runs could with ease be used to compare multigrid with the regular solvers. I would also suggest to plot the force network for each timestep and make it into an animation to see how it behaves under multigrid.

# References


[1] Algoryx Simulation AB. *AgX Dynamics*. Nov. 2013. URL: http://www.algoryx.se/.

[2] Olivier Pouliquen Bruno Andreotti Yoël Forterre. *Granular Media, Between Fluid and Solid*. 1st. Cambridge University Press, 2013.

[3] Nils Hjelte Claude Lacoursière. *Extended 2d tests partitioning and scheduling, Internal Document at Algoryx Simulation AB*. Jan. 2014.

[4] Spaceclaim corporation. Jan. 2014. URL: http://www.spaceclaim.com/.

[5] The HDF Group. *The HDF5 data model, library and file format*. Nov. 2013. URL: http://www.hdfgroup.org/HDF5/.

[6] C. T. Kelley. *Iterative Methods for Linear and Nonlinear Equations*. 2nd. Society for Industrial and Applied Mathematics, 1995.

[7] Claude Lacoursière. *Ghost and Machines: Regularized Variational Methods for Interactive Simulation of Multibodies with Dry Frictional Contacts*. 2007.

[8] The Lua programming language. Jan. 2014. URL: http://www.lua.org/.

[9] The Python programming language. Jan. 2014. URL: http://www.python.org/.

[10] P. Richard et al. "Slow relaxation and compaction of granular systems". In: *Nature Materials* 4 (Feb. 2005), pp. 121–128. DOI: 10.1038/nmat1300.

[11] Yousef Saad. *Iterative Methods for Sparse Linear Systems*. 2nd. Society for Industrial and Applied Mathematics, 2003.

[12] Martin Servin et al. *Examining the smooth and nonsmooth discrete element approach to granular matter*. 2013. URL: http://umit.cs.umu.se/granular/dem.

[13] Anton Schuller Ulrich Trottenberg Cornelis W. Oosterlee. *Multigrid*. Academic Press, 2000.

# Appendix A

# Derivations

This appendix will include some derivations from the theory.

## A.1 Derivations from the theory chapter

From equation 2.7 we have the general form $\mathbf{Ax} = \mathbf{b}$. If we take the matrix $\mathbf{A}$ and add/subtract another matrix to it, in this case the identity matrix $\mathbf{I}$, we get:

$$(\mathbf{A} + \mathbf{I} - \mathbf{I})\mathbf{x} = \mathbf{b} \tag{A.1}$$

Multiplying the matrices to $\mathbf{x}$ and moving two terms to the right-hand-side we get the following equation:

$$\mathbf{Ix} = -\mathbf{Ax} + \mathbf{Ix} + \mathbf{b} = (\mathbf{I} - \mathbf{A})\mathbf{x} + \mathbf{b} \tag{A.2}$$

Since multiplying the identity matrix with a vector gives the vector, $\mathbf{Ix} = \mathbf{x}$ in this case, which gives the start of the richardson iteration seen in equation 2.8