DYNAMIC TRIANGULATIONS FOR EFFICIENT 3D SIMULATION OF GRANULAR MATERIALS

THÈSE N^{O} 2432 (2001)

PRÉSENTÉE AU DÉPARTEMENT DE MATHÉMATIQUES

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Jean-Albert FERREZ

ingénieur mathématicien diplômé EPF

de nationalité suisse et originaire de Bagnes (VS)

acceptée sur proposition du jury:

Prof. Th. M. Liebling, directeur de thèse
Dr H.-R. Bircher, rapporteur
Prof. A. Mocellin, rapporteur
Dr D. Müller, rapporteur
Prof. A. Quarteroni, rapporteur
Dr M. Sawley, rapporteur

Lausanne, EPFL
2001

à Colette, Jean, Willy,
Pierre, Eliane et Jacques

Acknowledgements

Even though this thesis bears only my name, several persons were involved in its making.

My supervisor Tom Liebling is the kind of scientific and human leader that every young researcher wants to have. His extensive knowledge, his long experience, his numerous relations and involvements at EPFL and outside have opened many doors and allowed me to conduct this research in excellent conditions. Furthermore, I want to thank him above all for his unconditional respect of my decisions to focus on some new aspects or on the contrary to discard some of his many ideas.

This thesis is largely inspired from the pioneering work of Didier Müller, and I am grateful that he stopped at some point and left me one dimension to play with. Hansruedi Bircher provided inspiration and experimental data for packing spheres. I further thank Alain Mocellin, Alfio Quarteroni and Mark Sawley for taking part in the jury.

Claude Indermitte, Michel Bierlaire and Jean-François Hêche have provided portions of code for inclusion in my programs. Branko Glisic inspired section 6.2. Lionel Pournin helped me understand many of the physical aspects reported in chapter 3. Lionel and my brother Pierre have accepted the ungrateful task of proofreading the report. Several students have contributed to this thesis by accomplishing their semester or diploma projects under my supervision: Eliane Meichtry, Sandrine Paroz, Beatriz Andrade, Stefan Reinmann, Roland Berger, and especially Christophe Weibel who wrote a first draft of the parallel code.

Partial funding was provided by the Swiss National Science Foundation and the Swiss Defense Procurement Agency. I am grateful to both institutions for having made this research possible.

Working away from the mountains was balanced by the everyday pleasure of being part of the lively ROSO group. At the risk of forgetting someone, I would especially mention Alain Prodon, with whom I could improve my theoretical background while judging the benefits to *think different*; Michel Bierlaire, for the numerous scientific, technical, organizational and political lunch-time discussions; Christine Lütolf, for a very pleasant and motivating office atmosphere at MA128; Frank Crittin, for sharing hotel rooms around the globe, with or without luggage; and Noëlle Lieber, for being so efficient. I will also remember a very fruitful collaboration with our IT manager Jean-Claude Berney.

Last but not least, the amicable consideration from my friends – members of the *Ficelle Fan's Club* and others – and especially the logistical, financial, and loving support from my family have been essential driving forces throughout these years.

Abstract

Granular materials are omnipresent in many fields ranging from civil engineering to food, mining and pharmaceutical industries. Often considered a fourth state of matter, they exhibit specific phenomena such as segregation, arching effects, pattern formation, etc. Due to its potential capability of realistically rendering these behaviors, the Distinct Element Method (DEM) is a very enticing simulation technique. Indeed it makes it possible to analyze and observe phenomena that are barely if at all accessible experimentally. DEM works by tracking every particle in the system individually, maintaining for each a trajectory influenced by external factors such as gravitation or contacts with boundary objects and by the interactions with other grains.

The mathematical problem of identifying pairs of grains that interact and locating precisely where the contact occurs is highly dependent on the shape of the grains. We focus in this thesis on 3D spherical grains and use dynamic weighted Delaunay triangulations to track the collisions. The triangulation is built on the centers of the grains and evolves to follow their motion. We prove that all potentially colliding pairs of spheres are adjacent in the triangulation. As there are $6n$ to $8n$ edges for n spheres in most practical cases, the complexity of the collision detection becomes linear instead of quadratic in the number of particles, with a small overhead in maintaining the triangulation with efficient local operations.

For the physical problem of realistically rendering the collision in a numerical contact model suitable for computer simulation, we have used widely accepted theories such as the viscoelastic model of Cundall, but have also tested some recent, more sophisticated developments in the field.

The collision detection and contact models have been implemented in a modular DEM simulation code with advanced features in data structures storing the triangulation, in numerical robustness of the geometric computations, and in parallel processing on shared memory computers.

Optimal packing of powders is important in many industrial processes, yet no theoretical result exists when dealing with grains of different sizes. We have performed simulations of such cases and could compare our results with experimental data. Preliminary results have been obtained regarding the relation between the size and proportion of grains and the density of the packing.

Other simulations have also been performed, such as the granular flow through an hourglass. As no efficient simulation method is currently known for non-spherical 3D grains, we propose an intermediate approach of gluing spheres together into arbitrary shaped clusters and show some examples based on this approach.

Résumé

Les matériaux granulaires sont omniprésents dans plusieurs domaines tels que le génie civil, ou les industries agro-alimentaire, minière ou pharmaceutique. Souvent considérés comme un quatrième état de la matière, ils exhibent des phénomènes propres : ségrégation, effets d'arches et de surface, etc. De par son potentiel à reproduire de manière réaliste ces comportements, la méthode des éléments distincts (*Distinct Element Method*, DEM) est une technique de simulation attrayante. Elle rend en effet possible l'analyse et l'observation de phénomènes difficile voire impossible à obtenir expérimentalement. DEM fonctionne en suivant chaque particule du système indépendamment, tenant à jour pour chacune d'elles une trajectoire influencée par des facteurs externes tels que la gravité ou les contacts avec les parois, ainsi que par les interactions avec les autres grains.

Le problème mathématique consistant à identifier les paires de grains qui interagissent et à situer précisément le point de contact dépend fortement de la forme des grains. Nous considérons dans cette thèse des grains sphériques en 3D et utilisons une triangulation de Delaunay dynamique et pondérée pour détecter les collisions. La triangulation est construite sur les centres des grains et évolue de manière à suivre leurs mouvements. Nous prouvons que toutes les paires de sphères pouvant être en contact sont adjacentes dans la triangulation. Comme il y a en pratique entre $6n$ et $8n$ arêtes pour n sphères, la complexité de la détection des collisions devient linéaire au lieu de quadratique, avec un léger surcoût lié à la maintenance de la triangulation par des opérations locales efficaces.

Pour ce qui est du problème physique de rendre fidèlement le contact dans un modèle numérique destiné à la simulation, nous avons utilisé des théories universellement reconnues telles que le modèle viscoélastique de Cundall, mais nous avons également pu tester certains développements récents dans ce domaine.

La détection des collisions et les modèles de contact ont donné lieu à un code de simulation DEM modulaire ayant des catactéristiques avancées dans les structures de données pour la représentation de la triangulation, dans l'évaluation exacte et rapide des prédicats géométriques, et dans le calcul parallèle sur des machines à mémoire partagée.

Les mélanges de poudres jouent un rôle important dans plusieurs processus industriels, néanmoins aucun résultat théorique n'existe lorsqu'il s'agit de mélange de poudres de calibres différents. Nous avons simulé ces cas et pu comparer nos résultats à des données expérimentales. Nous avons ainsi obtenu des résultats préliminaires concernant la relation entre la taille et la proportion des divers grains d'une part et la densité du mélange d'autre part.

D'autres simulations ont été réalisées, telles que le flux granulaire dans un sablier. Comme aucune méthode efficace n'est connue à ce jour pour des grains non-sphériques en 3D, nous proposons une étape intermédiaire consistant à coller ensemble des sphères pour obtenir des amas de forme quelconque et présentons quelques exemples basés sur cette approche.

Contents

For every complex problem, there is one solution which is simple, neat and wrong.
– Henry Louis Mencken

Introduction

Improve. Upgrade. Revise. Develop. Enhance. Refine. Rationalize. Optimize. This constant need to be more efficient drives most economic and industrial activities. Whether to improve the quality of the products, to reduce toxic emissions or to satisfy greedy shareholders, engineers are constantly developing new manufacturing procedures, and mathematicians now have decades of experience in various optimization techniques. However, for optimization to be successful, indepth knowledge of the equipment and phenomena involved is necessary. Man has successfully build devices to accomplish certain tasks in a predictable manner, many aspects of our environment are controlled, but nature still has many secrets of its own to keep generations of scientists busy.

Ongoing efforts to understand the atomic structure of matter, the origin of the universe, the hereditary transmission of diseases, the encoding of the human genome, to cite but a few, are universally recognized and often make their way in the mainstream media.

In contrast, looking at a simple sand pile may sound like a flagrant lack of ambition. Can one think of a simpler element than a grain of sand? Is there some hidden collective intelligence in a heap? One is easily convinced that both questions call for negative answers, yet the actual behavior of the average sand pile, pepper pot, pill, grain elevator, box of corn flakes, or any other granular material one can think of, calls for better understanding.

Attempts were made to deduce a theoretical model for some aspects of granular materials, either by fitting these behaviors to that of better known solids, gas or liquids, or by original approachs. Experiments have been and are still performed to verify these models. However, computer simulations are increasingly seen as a means to not only replicate experimental measures but also to broaden the scope of situations that can be actually observed and measured.

As usual with numerical simulations of natural phenomena, a balanced mix of mathematics, physics, and computing is required to obtain an efficient and accurate model of reality. Granular materials are no exception. In recent years, a simulation technique that considers each grain in the system individually and lets them interact, the *Distinct Element Method* (DEM), has emerged as the most promising approach. The accuracy of DEM crucially hinges on the physical laws that drive the elements, while the efficiency requires not only powerful computers, but also clever algorithms to orchestrate the interactions.

In this thesis, we use an advanced structure of discrete geometry, the three-dimensional Delaunay triangulation, as the underlying model keeping track of the collective behavior of an assembly of spherical grains. Taking applied mathematics to the letter, we implemented this structure in a simulation code that allowed us to study several real-life problems, whenever possible in tight collaboration with scientists of the relevant fields.

Organization of this report

This thesis report is organized in two major parts, the first describes the methods and tools, and the second presents some applications.

Chapter 1 recalls the scientific environment of this work with a non-exhaustive enumeration of prior work and existing solutions in the major areas covered, from discrete mathematics (triangulations) to physics (numerical contact models) to computing science (parallelism).

Mathematical issues are dealt with in chapter 2 where we present and proove the major theoretical result of this thesis concerning the detection of collisions among spheres in 3D with dynamic triangulations. The classical notions of Voronoi diagrams and Delaunay triangulations are extended to weighted cases. We then explain how to maintain these structures in order to reflect the motion of the generating sites.

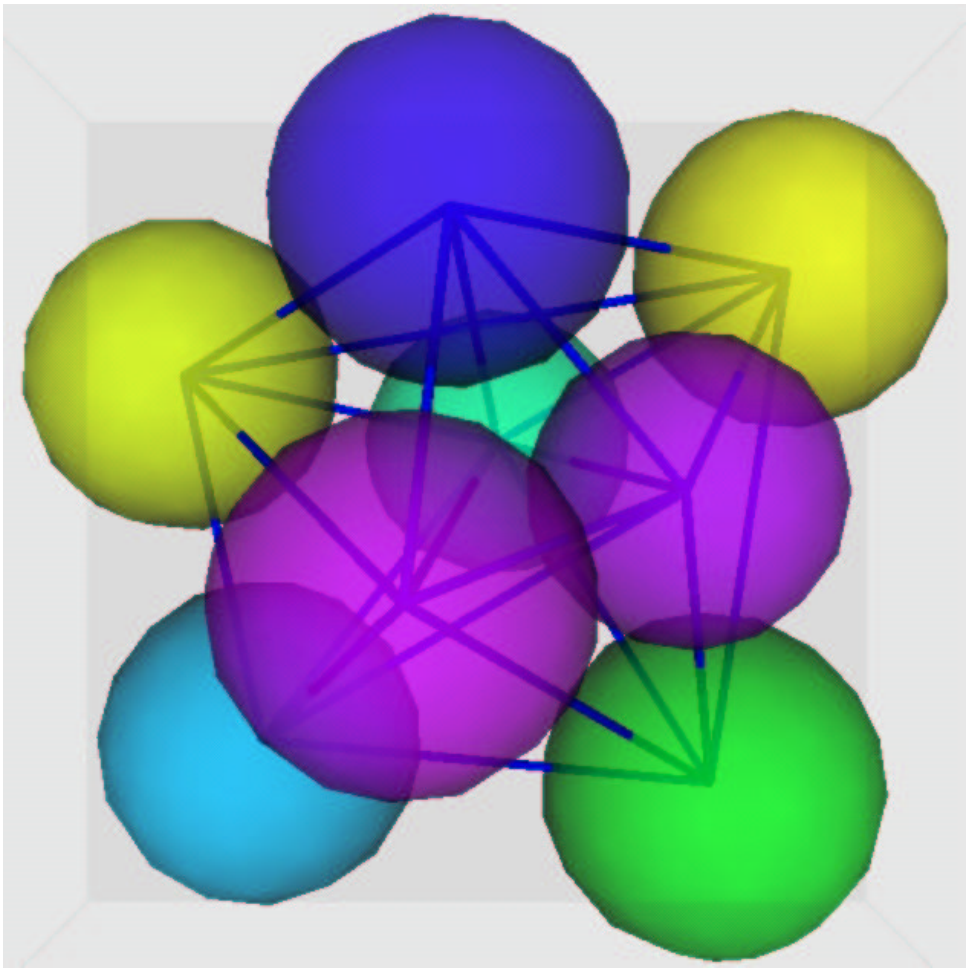
Chapter 3 adds a physical layer to this collision detection scheme by discussing the force models used to transform a geometric event — the collision of two spheres — into an accurate representation of a granular material. With all elements available, the complete simulation algorithm is presented.

Last but not least, we conclude the first part in chapter 4 by addressing some rather tedious implementation issues. Most of them come from the mathematical part, since the dynamic triangulation requires extreme care in the choice of data structures and the precision of the computation. The parallel version of the algorithm and some other minor issues are also discussed.

In the second part, chapter 5 presents the simulations we performed to study the packing of spheres of three different sizes. To our knowledge, it is the first time that this problem is tackled by computer simulation. This work was conducted in collaboration with scientists of the Energetic Material Labs of Swiss Defense Procurement Agency at Thun, who also provided us with results of experiments to which we could compare our simulations.

Several other applications are presented in chapter 6. These include the study of a sensor measuring deformations of concrete at early and very early age, and the granular flow in an hourglass. A new idea concerning grains formed by gluing spheres together is also presented.

This report, especially its second part, contains numerous color illustrations taken from the various simulations. Due to technical limitations, you may be looking at a black and white printed version in which most of those illustrations have lost a significant part of their impact. Should this be the case, you are encouraged to access a color version on the web at `http://rosowww.epfl.ch/jaf/3dwdt/`. Other technical limitations have prevented us to include animations of various phenomena in the present report. These animations are a natural complement of this thesis and are available as movies or three-dimentional models from the same web page.



Part I

Methods

Originality is the fine art of remembering what you hear
but forgetting where you heard it.
– Laurence J. Peter

Chapter 1

Building blocks and Motivation

The work we present in this thesis was achieved by integrating theories and concepts from various domains, mainly discrete geometry, numerical physics, and scientific computing. The next three chapters will focus on each of these aspects, but before that, we present here a selection of previous work that has inspired our developments. In particular, alternative collision detection methods, applications of triangulations, history of numerical contact models, and existing software in the field will be shortly reviewed.

1.1 Computer simulation of granular materials

Granular materials and flows are everywhere in nature, in various industrial processes, in everyday life. You find them in landslides and avalanches, erosion, raw minerals extraction and transport, cereal storage, powder mixing in chemistry or pharmaceutics, on your table as sugar, salt or pepper, just to cite a few examples. Granular materials are sometimes considered as a fourth state of matter, different from the classic solid, liquid and gas: they are not solid, because they do not resist stretching, but are not gaseous either, because they resist compression. And unlike liquids, their surface at rest needs not be flat. They exhibit specific phenomena that call for better understanding. To this end, experimental studies have been and are being conducted, but numerical simulation is increasingly seen as a means to understand and predict their behavior. Indeed, simulation has become a common tool in the design and optimization of industrial processes (see Cleary, 1998; Cleary and Sawley, 1999).

Attempts have been made to adapt standard fluid simulation methods to granular flows without much successes (see among others Jenkins and Savage, 1983). The continuous increase in computing power is now enabling researchers to implement more ambitious methods that do not focus on the granular assembly as an entity, but rather deduce its global characteristics from observing the individual behavior of each grain. This method is commonly referred to as the *Distinct Element Method* (DEM). Several variants exist, but two main issues are found in all of them: the localization of the collisions among the grains and the modeling of the actual contact between two grains. We will discuss the latter in section 1.4, but we first take a closer look at the collision detection.

Figure 1.1: Two examples of industrial processes involving granular materials: transportation by conveyor belt and storage.

1.2 Collision detection

A major algorithmic problem within DEM is the efficient detection of the interactions between the elements. Collision detection between objects has been studied for applications such as geometric modeling (Mantyla, 1988), computer graphics (Rogers, 1985; Feiner et al., 1990), robotics (Mirtich and Canny, 1995). The methods commonly used in DEM are in two stages. The first stage identifies the pairs of grains that can potentially be in contact at a given time by maintaining for every grain a list of neighbors. The second stage decides for every pair defined by the neighborhood whether or not there actually is contact, and in case there is locates it precisely on the grains involved.

Several neighborhood techniques have been used for the first stage, see Allen and Tildesley (1987) for a broad coverage of the topic. We mention briefly some of the most widely used, which are illustrated in figure 1.2. Verlet lists keep track of all neighbors within a given range, but are expensive to update[1]. Grid subdivisions allow only checking grains within a few adjacent cells, but require each cell to keep track of relevant grains. If the cells are small enough to contain at most one grain, the test in neighborhood cells is trivial, but the number of cells will be very large. If the cells are larger, collisions must be checked with all grains in the surrounding cells. Sigurgeirsson et al. (2000) discuss in great detail the optimal size of the cells with respect to the grain size, based on standard statistical assumptions.

These neighborhoods were not designed for populations of multiple sized grains. Adaptive grid subdivisions such as quadtrees (see Samet, 1984) can be used, but are more complex to update in order to reflect grain movements.

The second stage, the actual localization of the contact between two grains, is highly dependant on the shape of the grains. For spheres, the isotropic properties of the grains basically mean

[1]Furthermore, the range, well defined in the original context of Molecular Dynamics, is difficult to determine precisely since there is no notion of decreasing potential and cutoff distance in granular media simulations where forces only come from direct contacts.

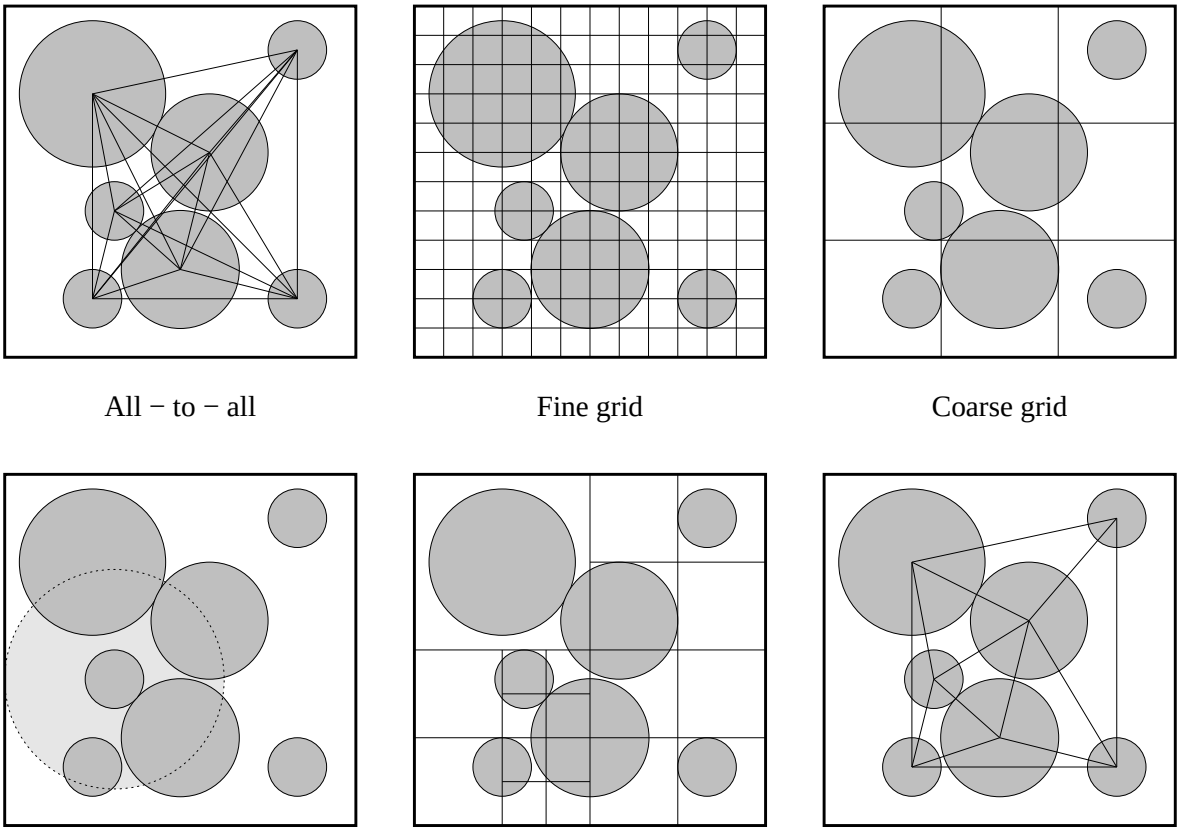


Figure 1.2: Various strategies for the first stage of the collision detection.

that there is nothing special to do. However, more complex geometries often require heavy algorithmic work: see Basch et al. (1999) for non-convex polyhedra; Erickson et al. (1999), Chung and Weng (1996), and Mirtich (1997) for convex polyhedra; Müller (1996a) for non convex polygons; O'Connor (1996) for superquadrics.

Since we deal only with spherical grains, only the first stage requires our attention. Müller (1996a) pioneered a new approach to efficient dynamic neighborhood in two dimensions by using triangulations. Among his ideas, at least the one concerning the discs could and should be extended to the three-dimensional case.

1.3 3D dynamic triangulations

2D and 3D dynamic triangulations have already been used as underlying structures in several modeling and simulation projects. Telley (1989) used 2D Laguerre complexes - the dual structure of the weighted Delaunay triangulation - to represent polycrystals and simulate normal grain growth in its long-term evolution. The model was then extended to 3D by Righetti (1992) and Xue (1995). Indermitte (1995) extended the 2D Delaunay triangulation to piecewise linear surfaces and used it to describe and model morphogenesis aspects of mycelial growth mech-

anisms. All these projects have been conducted in tight collaboration with researchers in the relevant field and resulted in validated models and efficient simulation schemes.

Müller (1996a) jumped into that train and used a triangulation based collision detection for distinct element simulations of granular media in two situations, see figure 1.3:

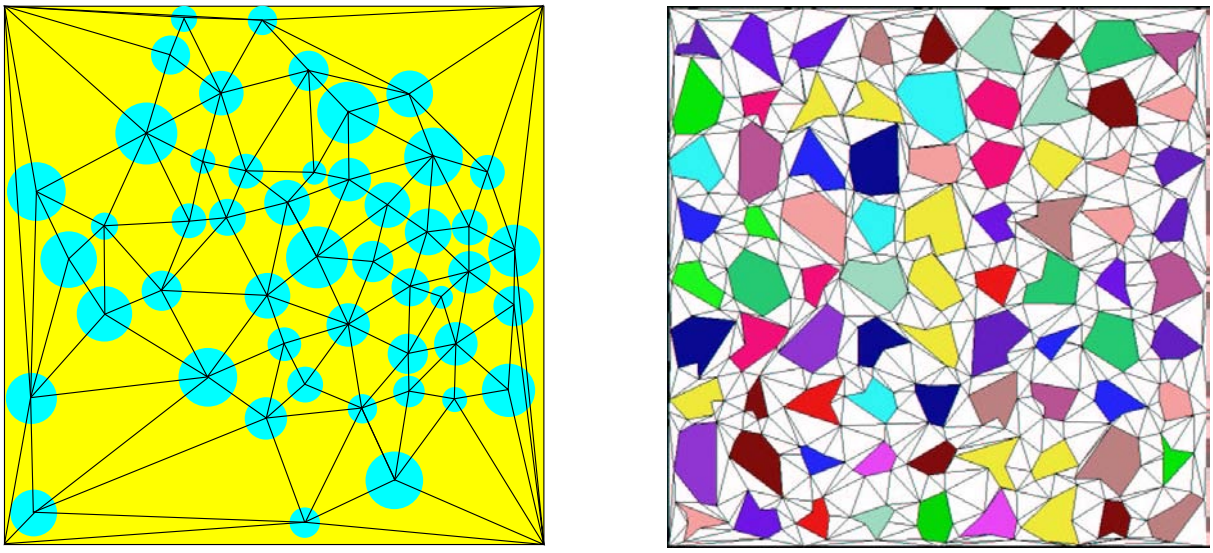


Figure 1.3: The neighborhoods used by Müller in 2D for discs and polygons.

- Grains represented by discs: In this case, a weighted Delaunay triangulation is built on the centers of the discs. Using the edges of the triangulation to detect possible contacts allowed to reduce the number of tests from $n(n-1)/2$ to $3n$, for n discs.
- Grains represented by polygons: A constrained triangulation is built in the interstices between the grains. Observing the dynamic evolution of the triangulation given the movements of the polygons allowed efficiently detecting the next collision and therefore compute for each grain an exact trajectory.

Müller concluded his thesis with the remark that the extension of those models to the three-dimensional case would be very interesting. But he also warned that the mathematical bases would require much more attention, that the computational implementation will be tedious, and that some of the grain interaction models might not be applicable anymore. In that sense, this thesis is the extension of his work.

Classically, the major application field of three-dimensional triangulations, and in particular the Delaunay triangulation, is mesh generation. Therefore, most attention was given to the creation of such triangulations, and not their evolution. Even the more recent trends in adaptive meshes do not qualify for the specific needs of DEM.

The dynamic behavior of the triangulation subject to the movement of the generating sites is the key issue for our purposes. Local transformations that maintain a given property while minimizing the changes to the triangulation are required. Such operations, called *flips*, were used by Joe (1989, 1991) as part of an insertion method to build the triangulation. Edelsbrunner and Shah

(1996) have a very comprehensive coverage of the flipping method for regular triangulations (i.e. weighted Delaunay) in E^d. Yet, none of those authors use the local operations to update a triangulation after some points have moved.

The weighted Delaunay triangulation works for our purpose because it is the dual structure of the Laguerre complex which exhibits the key properties we rely on, as we will explain in chapter 2. While Aurenhammer (1987) remains the reference in this area for much of the theory, Aurenhammer (1988) is an extension towards applications to some geometric problems. In particular, the sweeping procedure described by Hopcroft et al. (1983) is cited there as being more efficient than the Laguerre complex – Delaunay triangulation approach for collision detection among spheres in dimensions higher than two. While not in practice, this affirmation with the potential of obliterating our work is true in theory. The worst-case complexity of our scheme remains indeed in $O(n^2)$ because of degenerate cases (see section 2.5) while the method proposed by Hopcroft et al. has a complexity of $O(n \log n)$. However, even if the worst-case complexity speaks against us, practical experience has established that such cases are extremely rare and our linear scheme wins in practice. Furthermore, Hopcroft et al. only deal with individual collision detection and do nothing to continue tracking the collisions when the elements change position.

Guibas and Zhang (1998), as part of several results concerning *Kinetic Data Structures* (KDS) (see also Basch et al., 1997a,c,b), use the same tools as we do to solve a slightly different problem: they track the (single) pair of d-dimensional spheres that achieve the minimal distance, and prove that they are adjacent in the weighted Delaunay triangulation. Guibas et al. (1991) present the problem of queries among moving points and how the Voronoi can be used in this case as well. It is limited however to two-dimensional cases.

To close this brief review, let us mention two major reference books of the field by Sugihara et al. (1992) and Boissonnat and Yvinec (1995).

Once the contacts have been localized efficiently, the problem remains of doing something physically realistic with them.

1.4 Contact models

Two approaches have been adopted to update the trajectories of colliding grains. A first category of models, the *hard body* methods, act directly on the trajectories: whenever a collision is detected, the trajectories of the two grains are updated to reflect the situation immediately after the shock. This approach imposes individual and instantaneous collisions and is well suited for collision-oriented behaviors such as gases. They are *event driven*: the simulation is performed as a sequence of collisions. The second category of models, the *soft body* models, act on the forces that drive the elements: each collision produces a force, and the sum of all forces generates the acceleration, velocity and position of the elements. These models allow the grains to overlap and compute a force from the size and variation of these overlaps. They are continuous by nature, and the simulations based on them discretize time.

Numerical modeling of contacts between grains in a granular material must cover several situations: individual dynamic collisions, quasi-static motion, or static contacts. In order to repro-

duce all cases within the same model, soft body methods are often preferred to hard body methods, the latter suffering from inelastic collapse in dense situations, see McNamara and Young (1992).

The force computed by soft body models must be repulsive and dissipative. Repulsiveness is a result of the elasticity of the materials. Energy dissipation is a delicate topic as several mechanisms are responsible for the decrease of the total energy: plastic deformation (see Walton and Braun, 1986), noise (see Tejchman and Gudehus, 1993), temperature increase,...

For a single collision, those dissipative phenomena can be summarized by a coefficient of restitution, defined as the ratio of the relative velocities of the two grains after and before the collision (see Walton and Braun, 1986, and references therein for a complete collision theory). Ideally, this ratio would be chosen according to the real characteristics of the material, and the numerical model would render energy dissipation. Unfortunately, most models fail to do that accurately.

The history of numerical contact models begins in the late seventies with the linear viscoelastic model of Cundall and Strack (1979), and continues with non-linear viscoelastic models (Kuwabara and Kono, 1987), elasto-plastic models (Walton and Braun, 1986), hysteresis-based models... Most advances in the realism of these models come from the comparison of simple numerical simulations with the corresponding lab experiment (whenever such data is available) thus allowing designing and calibrating models.

1.5 Existing software

As most of the implementation effort for our work would be devoted to triangulations, we looked for available codes in this area. Software follows the same trend as the theory, with several mesh generation codes available. In particular, the FORTRAN implementation of the incremental procedure described in Joe (1991) is available. Mücke (1993) describes DETRI[2], a robust implementation for 3D Delaunay Triangulations. Other implementations, including parallel algorithms, can be found in Cignoni et al. (1993, 1995); Teng et al. (1993). Sugihara provides code for weighted Delaunay triangulations[3]. On the other hand, no efficient support for dynamic three-dimensional triangulation of moving points could be found until a few years back.

This situation is changing, as at least two libraries are being developed in the field: the *Library of Efficient Data structures and Algorithms*, LEDA[4] (see Mehlhorn and Näher, 1999) and the *Computational Geometry Algorithm Library*, CGAL[5]. When we started this project in 1997, none of them (LEDA 3.7 and CGAL 1.0) offered satisfactory support for 3D triangulations. A later attempt (Berger, 1999) to use LEDA v3.8 showed that the way they are implemented makes it difficult to deal with moving points. This situation has changed now, and especially CGAL offers some of the functionalities required in this context. The implementation of the

[2] Available at `http://www.geom.umn.edu/software/cglist/GeomDir/`.
[3] Available at `http://www.simplex.t.u-tokyo.ac.jp/~sugihara/opensoft/opensofte.html`.
[4] `http://www.algorithmic-solutions.com/`
[5] `http://www.cgal.org/`

DEM simulation loop and physical contact models on top of CGAL is thus left as an exercise for the reader.

1.6 Parallel computing

Realistic DEM simulations require large sets of elements and therefore high performance computing (HPC) infrastructure. Over the last ten years or so, HPC has increasingly involved some form of parallel computing where the total work is split among several processors. With EPFL being the first non-US site to receive a Cray T3D massively parallel machine in 1994, it was tempting to adapt the 2D simulations of Müller on that machine. This was done a year later when 2D rockfall simulations with 20'000 grains were performed more than 30 times faster on that machine than on a standard workstation (see Ferrez et al., 1996). The Cray T3D was a fast machine, but its potential was difficult to exploit. Single CPU performance was relatively poor, because of the absence of second level cache. On the other hand, an efficient scheme (`shmem`, see Barriuso and Knies, 1994) provided transparent access to the total distributed memory, which enabled us to parallelize the code quickly and efficiently. The next generation of parallel machines, represented at EPFL by the SGI Origin2000, brought additional performance and the ease of use of a globally shared memory. However, experiments performed for another project (see Ferrez et al., 1998a,b) showed that the shared memory advertised by this machine did not deliver the expected performance and that the algorithms needed to be adapted for the underlying distributed memory in order to achieve good results (see also Chandra et al., 1997).

At about the same time, low cost PCs based on Intel Pentium processors began to threaten the dominance of expensive RISC-based workstation in terms of processing power[6]. This trend was enforced by the worldwide emergence of computing farms built with hundreds or thousands of PCs connected with traditional networking equipment and running Linux. Those systems are commonly designed as beowulf[7]. The EPFL has also followed this trend, even though it took a biased approach. The Swiss-Tx project (see Kuonen and Gruber, 1999, and `http://capawww.epfl.ch/swiss-tx/`), in which the author took a minor part, built machines based on Alpha processors running a proprietary Unix and devoted a major part of its resources to the development of custom high performance networking hardware and software.

[6] See for example a brief comparison made by the author using the HINT benchmark of Gustafson et al. (1999) available at `http://rosowww.epfl.ch/jaf/HINT/`.

[7] See `http://www.beowulf.org/` for lists of such systems worldwide.

In mathematics you don't understand things.
You just get used to them.
– Johann Von Neumann

Chapter 2

Three-dimensional dynamic triangulations

2.1 Introduction

Our collision detection scheme is based on a three-dimensional dynamic weighted Delaunay triangulation such as the one shown in figure 2.1. This chapter is devoted to a presentation of this geometric data structure and its properties. In particular, we show that this scheme works as advertised, in the sense that under very mild hypothesis no collisions are missed. This chapter addresses only the theoretical issues, chapter 3 will insert the detection scheme in the global DEM framework, and chapter 4 will join theory and practice by discussing various implementation details.

We introduce the concept of Laguerre cells, also known as *power cells*, and the resulting Laguerre complex or *power diagram*. They have been extensively studied by Aurenhammer (1987) and used by Telley et al. (1996a,b) and Xue et al. (1997). We also use their dual structure, the weighted Delaunay triangulation or *regular triangulation* for which important results are due to Edelsbrunner and Shah (1996). We recall their definitions and mention some of their properties that are useful to our purpose, but the reader is referred to those papers and references therein for a thorough discussion. Boissonnat and Yvinec (1995) also have a very complete coverage of those issues.

We assume in a first approach that the centers of the grains are in *general position*: no three of them are collinear, no four of them are coplanar and no five of them are cospherical[1]. This simplifies the discussion by asserting the existence and uniqueness of some concepts. Then in section 2.5 we shall discuss in greater detail the implications of this hypothesis. In particular, we shall see that violating this condition only for short periods of time does not hinder our objective.

We focus on the three-dimensional case because our application is modeling and simulation of 3D granular media. For clarity's sake, illustrations will show a two-dimensional situation whenever possible.

[1]These are the standard criteria of non-degeneracy. Since we use weighted points the criteria are slightly different.

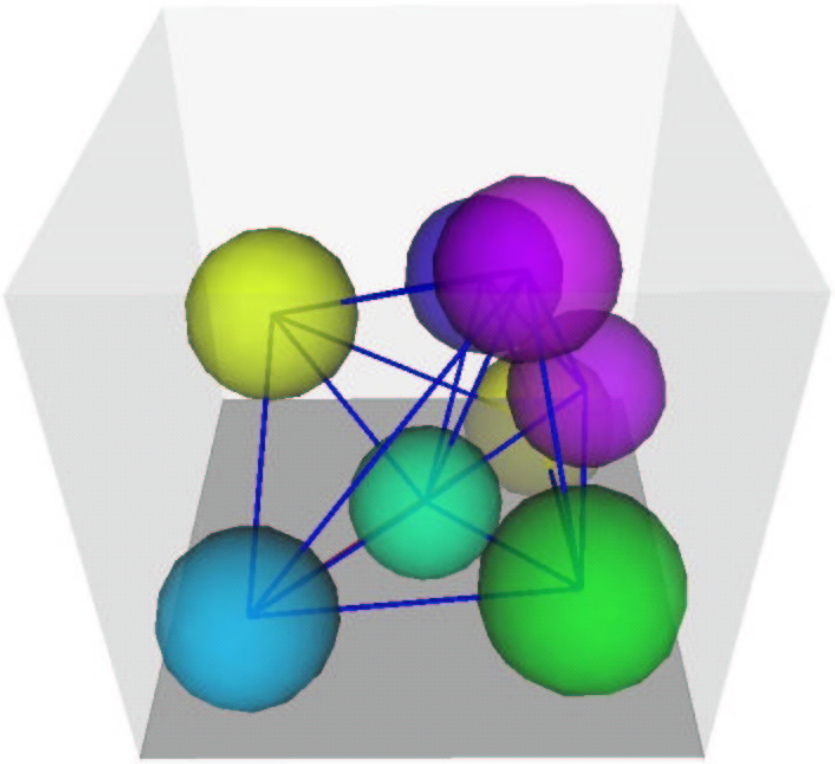


Figure 2.1: Some almost translucent spheres in a cubic box showing the edges of the weighted Delaunay triangulation used to detect potential contacts among them.

2.2 Static case

2.2.1 A set of grains

Consider a set $\mathcal{K}$ of n spherical grains G_i, $i = 1 \dots n$, in three-dimensional Euclidean space E^3 where grain G_i has center $\vec{c}_i$ and radius r_i. In typical applications n will be relatively large, in the 10^5 to 10^8 range. No assumption is made on the distribution of the radii r_i. The spatial distribution of the centers is expected — but not required — to be relatively compact, that is the grains fill some arbitrarily shaped container with a relatively high density.

From the point of view of geometry, the grains we consider are really balls, in that they are the set of points at distance less or equal than r from the center $\vec{c}$: $G_i = \left\{ \vec{x} \in E^3 \mid \|\vec{x} - \vec{c}_i\| \leq r_i \right\}$. Occasionally we refer to the surface or border of G, the set of points at distance exactly r from $\vec{c}$ that we note $\overline{G}$: $\overline{G}_i = \left\{ \vec{x} \in E^3 \mid \|\vec{x} - \vec{c}_i\| = r_i \right\}$, and the interior of G, the set of points at distance less than r from $\vec{c}$ that we note $\underline{G}_i = \left\{ \vec{x} \in E^3 \mid \|\vec{x} - \vec{c}_i\| < r_i \right\} = G_i \setminus \overline{G}_i$

The grains are allowed to overlap slightly, thus allowing the use of soft particle contact models where the physics of the shock is simulated by a small deformation of the grain (see chapter 3), resulting in overlaps no larger than a few percent of the diameters. If overlaps larger than half the radius of the smaller grain are encountered, then this method cannot be used anymore. This restriction on the size of the overlaps also guarantees that at least some portion of each grain is

not covered by any other grain or, in other words, that no grain is completely included in the union of other grains, see figure 2.2.

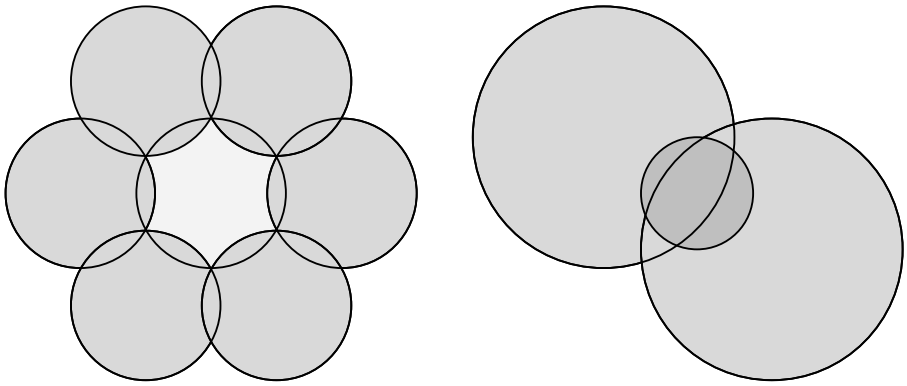


Figure 2.2: The situation on the left is allowed: the light grain in the center is completely surrounded, but not completely covered. The situation on the right is forbidden: the dark grain is completely covered by the two others.

Thus, we distinguish three cases for the relative positions of two given grains G_i and G_j, see figure 2.3. If $|G_i \cap G_j| = 0$, they are *disjoint*, if $|G_i \cap G_j| = 1$, they *touch*, and if $|G_i \cap G_j| > 1$, they *overlap*. We say that two grains *collide* if they touch or overlap.

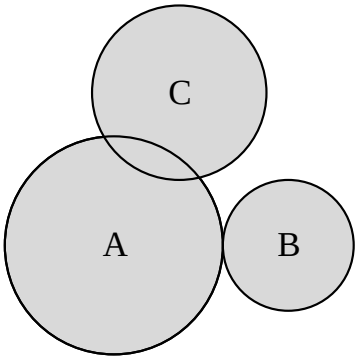


Figure 2.3: Relative positions of grains: B and C are disjoint, A and B touch, A and C overlap. The collisions to be detected are between A and B and between A and C.

2.2.2 The power function with respect to a grain

Given a grain G and a point x in E^3, the *power function of x with respect to G* is given by

$$P(\vec{x}, G) = \|\vec{c} - \vec{x}\|^2 - r^2 \tag{2.1}$$

see figure 2.4. We will use the following simplified notation

$$P_i(\vec{x}) = P(\vec{x}, G_i) \tag{2.2}$$

whenever there is no ambiguity.

We associate a weight w_i to grain G_i with $w_i = r_i^2$, thus sometimes referring to the couples $(\vec{c}_i, w_i)$ as weighted sites. $P_i(\vec{x})$ is then referred to as the power distance to the (weighted) site $\vec{c}_i$.

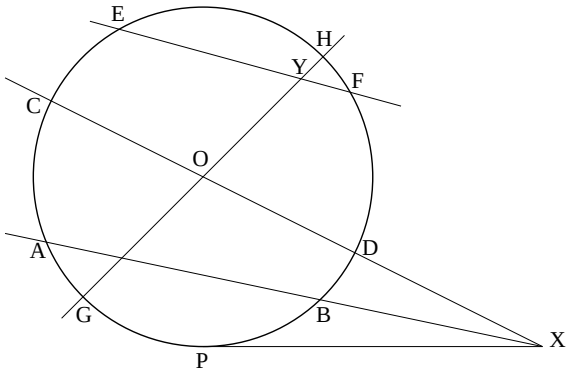


Figure 2.4: The power function of a point X *outside* the disk is defined as the dot product $XA \cdot XB$ for any choice of the (collinear) points A and B on the boundary. In particular, it is equal to $XP \cdot XP = \|XP\|^2$, thus > 0, and to $XC \cdot XD = (\|XO\| + r)(\|XO\| - r) = \|XO\|^2 - r^2$. The power function of a point Y *inside* the disk is also defined as $YE \cdot YF$, thus < 0, and in particular equal to $YG \cdot YH = -(r + \|YO\|)(r - \|YO\|) = -(r^2 - \|YO\|^2) = \|YO\|^2 - r^2$. These definitions carry over to three dimensions where the disk becomes a ball.

2.2.3 The Laguerre complex

To each grain G_i we associate a portion of E^3 called the *Laguerre cell* L_i (see figure 2.5) defined as:

$$L_i = \left\{ \vec{x} \;\middle|\; P_i(\vec{x}) \leq P_j(\vec{x}) \quad \forall j \neq i \right\} \tag{2.3}$$

The center of G_i $\vec{c_i}$ is the generating *site* of L_i. This definition is similar to that of the well-known Voronoi cell. It essentially means that L_i contains all the points that are "closer" to G_i than to any other grain, but using the power function with respect to a grain as a "distance". It can be seen as a weighted extension of the Voronoi cell. The two are equivalent if all the weights w_i are equal, in our case if all the grains have the same size.

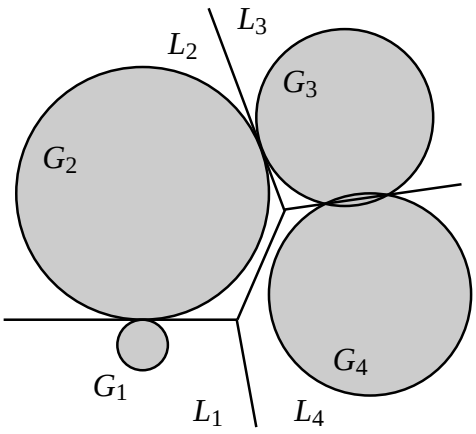


Figure 2.5: The grains G_i and their Laguerre cells L_i.

Each Laguerre cell is a convex polyhedron. Voronoi cells of distinct points are never empty, but Laguerre cells can be empty for some values of the weights. However, our assumption that a

grain may not be completely covered enforces the non-emptiness of our cells: For each grain G_i there exists at least one point $\vec{p}_i \in \underline{G_i}$ such that $\vec{p}_i \notin G_j \quad \forall j \neq i$. Thus, by definition of the power function,

$$P_i(\vec{p}_i) < 0 \quad \text{and} \quad P_j(\vec{p}_i) > 0 \quad \forall j \neq i \tag{2.4}$$

and L_i contains at least $\vec{p}_i$. The union of these cells is the entire space. If the centers of the grains are in general position, the intersection of two cells L_i and L_j is either empty or a convex polytope contained in the chordal plane orthogonal to $(\vec{c}_i\vec{c}_j)$. Figure 2.6 shows the position of this chordale. For the set $\mathcal{K}$ of grains G_i, the Laguerre cells L_i yield a partition of E^3 called the Laguerre complex $\mathcal{LC}(\mathcal{K})$, see again figure 2.5.

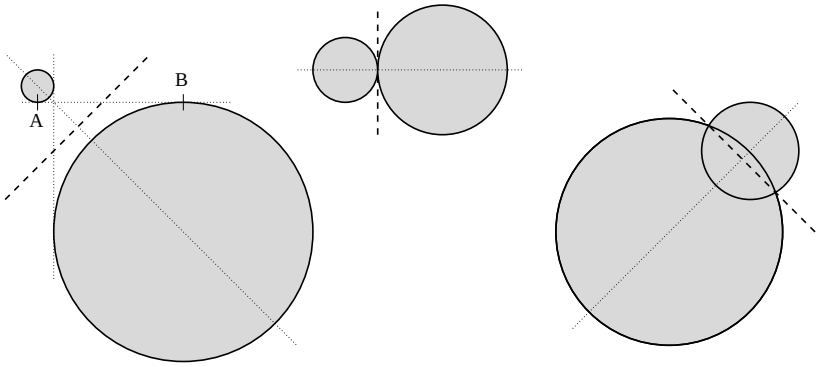


Figure 2.6: The thick dashed line is the chordale between two adjacent Laguerre cells. In the disjoint case (left), it splits the segment AB in two equal parts.

If a grain G_i touches or overlaps none of the others, it is entirely contained in its Laguerre cell L_i. If two grains G_i and G_j touch at a point $\vec{x_{ij}}$, then we have $P_i(\vec{x_{ij}}) = P_j(\vec{x_{ij}}) = 0$ and therefore $\vec{x_{ij}} \in L_i$ and $\vec{x_{ij}} \in L_j$ and the two Laguerre cells L_i and L_j have a non-empty intersection. If G_i and G_j overlap, the set of points $\vec{x_{ij}}$ such that $P_i(\vec{x_{ij}}) = P_j(\vec{x_{ij}}) = 0$ is exactly $\overline{G_i} \cap \overline{G_j}$ and is a circle in the chordal plane. There are also points $\vec{p_{ij}}$ such that $P_i(\vec{p_{ij}}) = P_j(\vec{p_{ij}}) < 0$, these points are in $\underline{G_i} \cap \underline{G_j}$. The two Laguerre cells L_i and L_j have again a non-empty intersection.

Thus, detecting collisions may be done only for the pairs of grains for which the corresponding Laguerre cells have a non-empty intersection.

2.2.4 The Delaunay triangulation

The Laguerre complex $\mathcal{LC}(\mathcal{K})$ has a dual structure, called the weighted *Delaunay triangulation* $\mathcal{DT}(\mathcal{K})$, see figure 2.7. Under the same hypothesis about the grains being in general position, it is uniquely defined: $\vec{c}_i$ and $\vec{c}_j$ are adjacent in $\mathcal{DT}(\mathcal{K})$ if and only if L_i and L_j have a non-empty intersection in $\mathcal{LC}(\mathcal{K})$. This finally brings us to our property:

Detecting colliding grains in $\mathcal{K}$ need only be done among the pairs that are adjacent in $\mathcal{DT}(\mathcal{K})$.

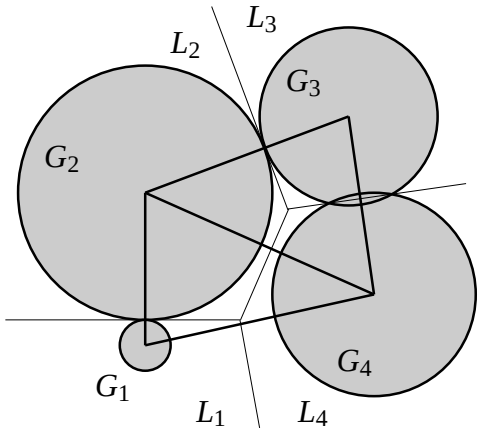


Figure 2.7: The grains G_i, their Laguerre cells L_i (thin lines), and the weighted Delaunay triangulation (thick lines) on the center of the grains.

If edge (i, j) is not in the triangulation, then the grains are disjoint. Note that the converse is not true, as there are edges in the triangulation between pairs of disjoint grains. This is an overhead of the method, but it is reasonable: Instead of the $n(n-1)/2$ possible pairs, the number of edges in a 3D weighted Delaunay triangulation is O(n) if the centers $\vec{c}_i$ are in general position. Typical 3D granular media simulations result in $\mathcal{DT}$ having between 6n and 8n edges.

2.3 Geometric predicates

The triangulation is built on the centers of the grains, and thus every vertex of the triangulation is associated with a point in Euclidean space E^3. However, we are not really using this continuous geometric information for the collision detection[2]: only the discrete adjacency information, that is the pairs of grains identified by edges of the triangulation, is useful for our purpose. In other words, we are interested in the topology of the triangulation, and not its actual shape. Nevertheless, this topology is completely determined by the geometric information contained in the coordinates (and weights) of the generating sites. The abstraction step from the continuous (and continuously changing, as we will see in the next section) coordinates of the grains to the discrete topology of the triangulation represented by the set of edges is provided by the weighted version of the *insphere* predicate.

A geometric predicate is a function of real parameters that returns a result in the set {-1, 0, 1}. From the continuous coordinates of a set of points it produces a discrete left-center-right or inside-on-outside answer. One of the simplest examples is the two-dimensional orientation predicate: given three points A, B and P in E^2, it will decide whether P is on the left or the right of the (directed) line AB, and the special case where the three points are collinear is also detected. Another example is the incircle predicate that decides whether a point P lies inside, outside or exactly on the circle defined by three non-collinear points A, B and C. Those two bi-dimensional predicates are illustrated in figure 2.8.

[2] At least not for the first phase, the definition of a neighborhood. The actual localization of the contact point (pahse 2) obviously requires the exact coordinates

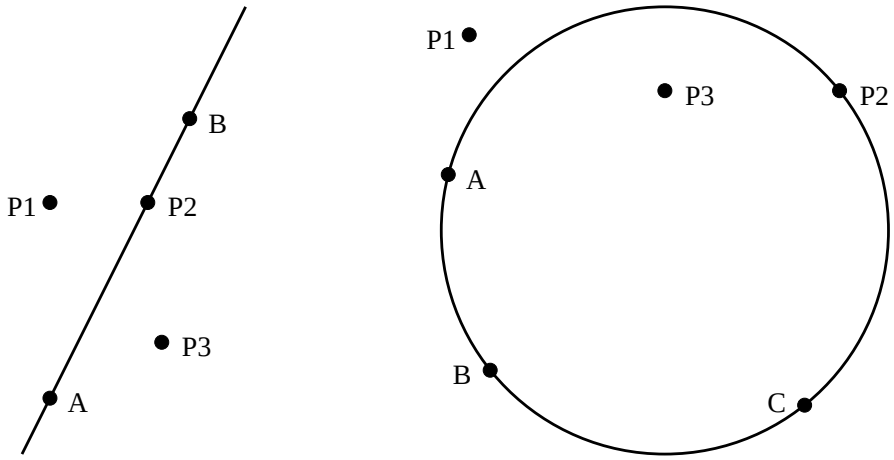


Figure 2.8: The Orient2D and Incircle2D predicates.

The predicates often reduce to the computation of the sign of a matrix determinant, yielding a positive or negative answer, with zero being the special value resulting from a degenerate imput. Two predicates are heavily used for the three-dimensional weighted Delaunay triangulation. Orient3D and WeightedInsphere3D.

2.3.1 Orient3D

Given three non-collinear points A, B and C in E^3, the Orient3D predicate answers the following question: on which side of the hyperplane defined by A, B and C does a point P lie ? Possible answers are left or right, the orientation being decided by the orientation of the triangle ABC, or again the special degenerate case if the four points are coplanar.

The Orient3D predicate is computed as the sign of the volume of the oriented tetrahedron built on the four input points. Elementary geometry and linear algebra yield the following equivalent expressions:

$$\mathrm{Or3D}(\mathrm{A},\mathrm{B},\mathrm{C},\mathrm{D}) = sign \begin{vmatrix} \mathrm{A}_x & \mathrm{A}_y & \mathrm{A}_z & 1 \\ \mathrm{B}_x & \mathrm{B}_y & \mathrm{B}_z & 1 \\ \mathrm{C}_x & \mathrm{C}_y & \mathrm{C}_z & 1 \\ \mathrm{D}_x & \mathrm{D}_y & \mathrm{D}_z & 1 \end{vmatrix} \tag{2.5}$$

$$= sign \begin{vmatrix} \mathrm{A}_x - \mathrm{D}_x & \mathrm{A}_y - \mathrm{D}_y & \mathrm{A}_z - \mathrm{D}_z \\ \mathrm{B}_x - \mathrm{D}_x & \mathrm{B}_y - \mathrm{D}_y & \mathrm{B}_z - \mathrm{D}_z \\ \mathrm{C}_x - \mathrm{D}_x & \mathrm{C}_y - \mathrm{D}_y & \mathrm{C}_z - \mathrm{D}_z \end{vmatrix} \tag{2.6}$$

The order of the four points obviously plays a role. The sign of the determinant changes when rows are permuted, but these changes correspond to the changes of geometric orientation.

The Orient3D predicate is used when we create the initial triangulation, as described in §4.2.1.

2.3.2 Insphere3D

Four non-coplanar points A, B, C and D in E^3 define a unique sphere. The insphere predicate answers the following question: Does a point P lie inside or outside that sphere ? Possible answers are inside or outside, or again the special degenerate case if the five points are cospherical.

In §2.2.4 we have introduced $\mathcal{DT}(\mathcal{K})$ as the dual of $\mathcal{LC}(\mathcal{K})$, but the Delaunay triangulation has some remarkable properties on its own. Among all possible triangulations it is the only one satisfying the insphere[3] criterion: the sphere defined by the four vertices of every tetrahedron of $\mathcal{DT}(\mathcal{K})$ contains no other points of $\mathcal{K}$. Figure 2.9 illustrates this property in 2D. It extends to the weighted Delaunay triangulation by using a weighted version of the insphere criterion, see below.

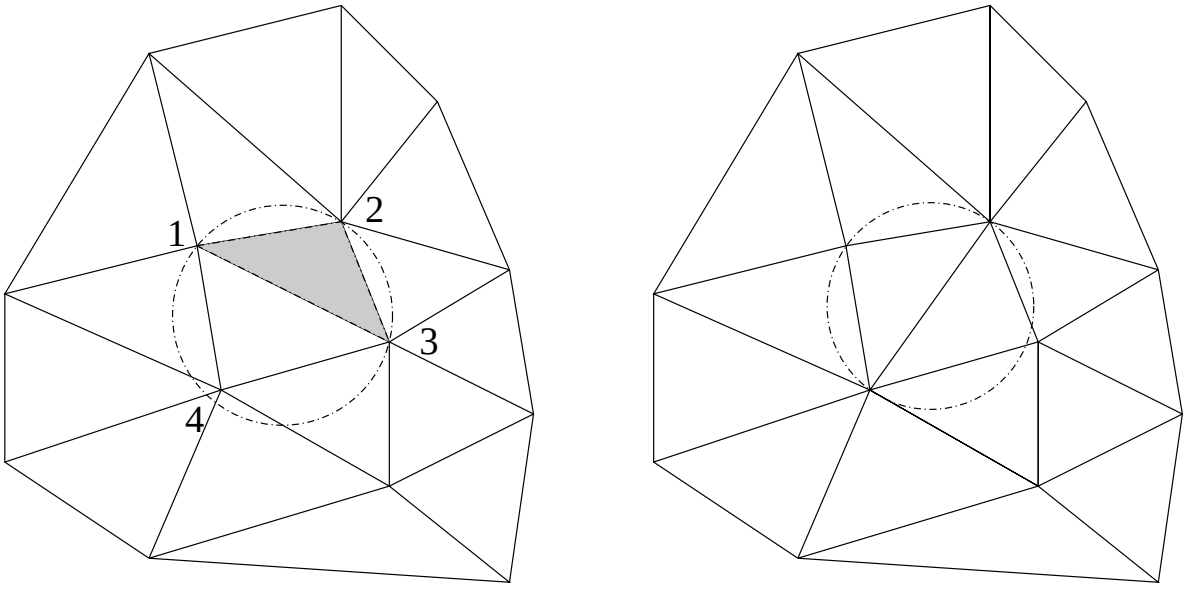


Figure 2.9: The incircle criterion: in the left triangulation, the gray triangle is not valid. The slightly different triangulation on the right is valid. It is the Delaunay triangulation.

One possible implementation for the insphere predicate is to explicitly compute the center Z of the sphere and then compare the distances AZ and PZ. A more efficient approach is to write implicitly the equation of the sphere using the coordinates of A, B, C and D as parameters and those of P as variables. We give here a different method.

We add one more dimension and project the points on a paraboloid in E^4: A, of coordinates (A_x, A_y, A_z) is projected onto A^+ of coordinates (A_x, A_y, A_z, $A_x^2+A_y^2+A_z^2$) and likewise for B, C, D and P. The four points A^+, B^+, C^+ and D^+ define a hyperplane in E^4. If P is in the sphere, then its projection on the paraboloid P^+ will be below the hyperplane. Thus, the three-dimensional insphere predicate may be implemented as a four-dimensional orientation predicate and we fall

[3]There is an unfortunate name clash here. The term insphere refers to the the property of a given point to be inside or outside a sphere, and not to the choice of the sphere with respect to the tetrahedron.

back on a volume sign computation. Figure 2.10 illustrates this concept for the one-dimensional case.

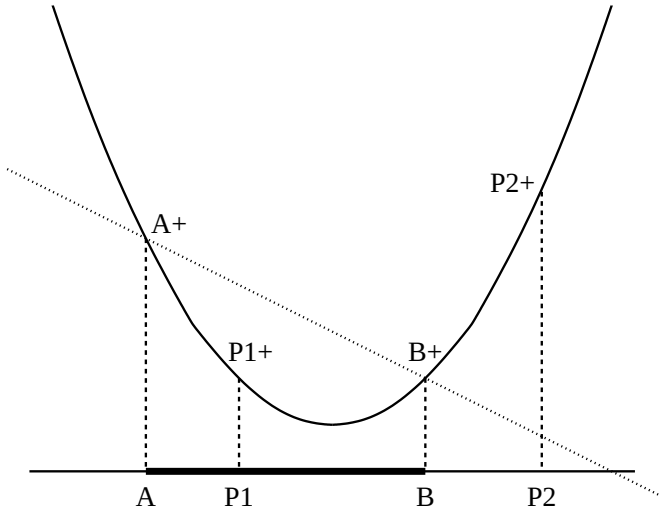


Figure 2.10: The one-dimensional insegment predicate. P1 is inside the segment AB because $P1^+$ is below A^+B^+ while P2 is outside because $P2^+$ is above A^+B^+.

$$\begin{aligned} \mathrm{InSph3D}(A,B,C,D,E) &= \mathrm{Or4D}(A^+,B^+,C^+,D^+,E^+) \\ &= sign \begin{vmatrix} A_x & A_y & A_z & A_x^2+A_y^2+A_z^2 & 1 \\ B_x & B_y & B_z & B_x^2+B_y^2+B_z^2 & 1 \\ C_x & C_y & C_z & C_x^2+C_y^2+C_z^2 & 1 \\ D_x & D_y & D_z & D_x^2+D_y^2+D_z^2 & 1 \\ E_x & E_y & E_z & E_x^2+E_y^2+E_z^2 & 1 \end{vmatrix} \end{aligned} \tag{2.7}$$

$$= sign \begin{vmatrix} A_x-E_x & A_y-E_y & A_z-E_z & (A_x^2+A_y^2+A_z^2)-(E_x^2+E_y^2+E_z^2) \\ B_x-E_x & B_y-E_y & B_z-E_z & (B_x^2+B_y^2+B_z^2)-(E_x^2+E_y^2+E_z^2) \\ C_x-E_x & C_y-E_y & C_z-E_z & (C_x^2+C_y^2+C_z^2)-(E_x^2+E_y^2+E_z^2) \\ D_x-E_x & D_y-E_y & D_z-E_z & (D_x^2+D_y^2+D_z^2)-(E_x^2+E_y^2+E_z^2) \end{vmatrix} \tag{2.8}$$

Incidentally, equation (2.7) is one way to write the equation of the sphere defined by A, B, C and D for the variables E_x, E_y and E_z since it is of the form $E_x^2+E_y^2+E_z^2+\alpha E_x+\beta E_y+\gamma E_z+\delta=0$ and is verified by all four points A, B, C and D.

2.3.3 WeightedInsphere3D

To account for the weighted case, one must first determine the sphere defined by four weighted points in E^3. In the previous section, we introduced the Laguerre complex as the weighted extension of the Voronoi diagram using the power function as a "distance". It seems reasonable to consider a point in E^3 whose power functions to all four weighted points A, B, C and D are

equal as the center of that sphere. Then, to decide whether the fifth weighted point is inside, on, or outside the sphere one uses again the power function.

Furthermore, since the weights of our points are the square of the radii of the spherical grain they represent, we can use the notion of orthogonal spheres: spheres S_1 of center $\vec{c}_1$ and radius r_1 and S_2 of center $\vec{c}_2$ and radius r_2 are orthogonal if

$$\|\vec{c}_2 - \vec{c}_1\|^2 = r_1^2 + r_2^2 \tag{2.9}$$

or equivalently if

$$P_1(\vec{c}_2) = r_2 \quad \text{or} \quad P_2(\vec{c}_1) = r_1 \tag{2.10}$$

see figure 2.11.

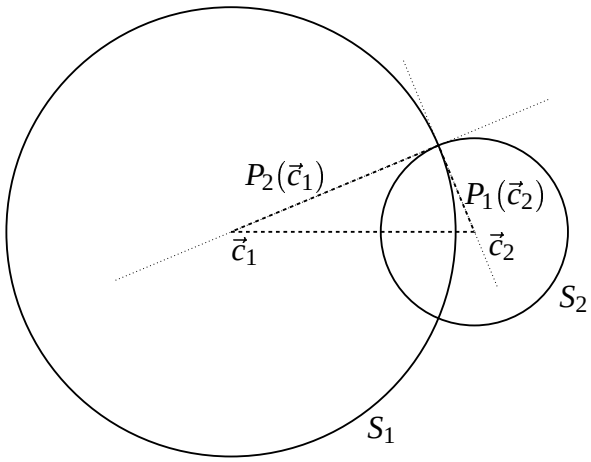


Figure 2.11: Two orthogonal spheres.

The sphere defined by four grains used for the weighted insphere predicate in 3D is the unique sphere orthogonal to all four grains[4], see figure 2.12.

We have the criterion, it remains now to actually write an expression similar to equation (2.7) for the weighted case. One can achieve it by rewriting the equation of the sphere using the power function as the distance. We will proceed differently, with yet another property of the Delaunay and weighted Delaunay triangulations. Consider the set $\mathcal{K}$ of points in E^3 and lift them again on the paraboloid in E^4, thus obtaining the set $\mathcal{K}^+$. Consider now the edges of the convex hull of $\mathcal{K}^+$. Looking from above, only exterior edges are visible and once projected back on the original hyperspace E^3 they correspond to the edges of the convex hull of $\mathcal{K}$. Looking from below, all edges are visible as the points of $\mathcal{K}^+$ are on the paraboloid. Those edges, once projected back on the original hyperspace E^3 are exactly the edges of the Delaunay triangulation $\mathcal{DT}(\mathcal{K})$.

This extends to the weighted case if instead of lifting point A on the paraboloid, that is at an altitude of $A_x^2 + A_y^2 + A_z^2$, it is lifted at a slightly lower altitude of $A_x^2 + A_y^2 + A_z^2 - A_w$, where A_w is the weight of A. Using again the orient4D predicate, we deduce from this property the following expressions for the WeightedInsphere3D predicate:

[4]Such a sphere is unique, but may not exist. If it does not exist, we have the weighted equivalent of four coplanar points, and the weighted sites are not in general position.

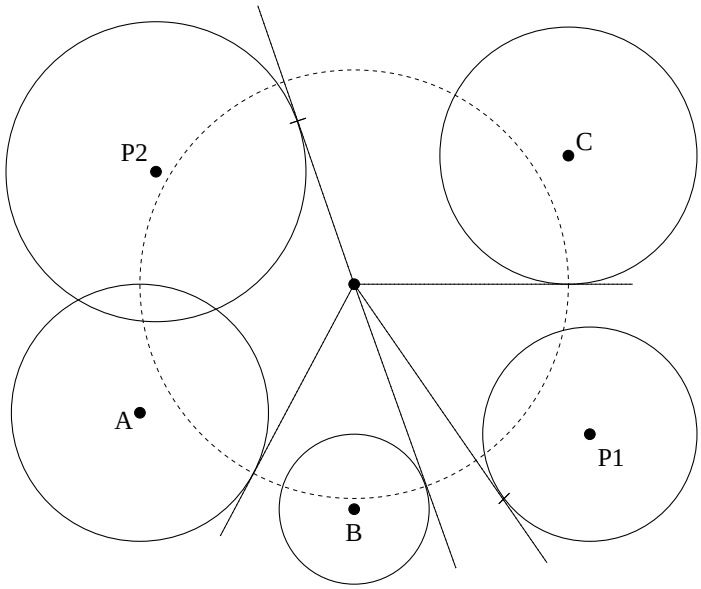


Figure 2.12: The weighted incircle predicate in 2D: the dashed circle is orthogonal to the circles representing the weighted points A, B and C. P1 is "outside" while P2 is "inside".

$$\mathrm{WInSph3D}(\mathrm{A},\mathrm{B},\mathrm{C},\mathrm{D},\mathrm{E}) \;=\; \mathit{sign}\begin{vmatrix} \mathrm{A}_x & \mathrm{A}_y & \mathrm{A}_z & (\mathrm{A}_x^2+\mathrm{A}_y^2+\mathrm{A}_z^2)-\mathrm{A}_w & 1 \\ \mathrm{B}_x & \mathrm{B}_y & \mathrm{B}_z & (\mathrm{B}_x^2+\mathrm{B}_y^2+\mathrm{B}_z^2)-\mathrm{B}_w & 1 \\ \mathrm{C}_x & \mathrm{C}_y & \mathrm{C}_z & (\mathrm{C}_x^2+\mathrm{C}_y^2+\mathrm{C}_z^2)-\mathrm{C}_w & 1 \\ \mathrm{D}_x & \mathrm{D}_y & \mathrm{D}_z & (\mathrm{D}_x^2+\mathrm{D}_y^2+\mathrm{D}_z^2)-\mathrm{D}_w & 1 \\ \mathrm{E}_x & \mathrm{E}_y & \mathrm{E}_z & (\mathrm{E}_x^2+\mathrm{E}_y^2+\mathrm{E}_z^2)-\mathrm{E}_w & 1 \end{vmatrix} \tag{2.11}$$

$$= \mathit{sign}\begin{vmatrix} \mathrm{A}_x-\mathrm{E}_x & \mathrm{A}_y-\mathrm{E}_y & \mathrm{A}_z-\mathrm{E}_z & (\mathrm{A}_x^2+\mathrm{A}_y^2+\mathrm{A}_z^2-\mathrm{A}_w)-(\mathrm{E}_x^2+\mathrm{E}_y^2+\mathrm{E}_z^2-\mathrm{E}_w) \\ \mathrm{B}_x-\mathrm{E}_x & \mathrm{B}_y-\mathrm{E}_y & \mathrm{B}_z-\mathrm{E}_z & (\mathrm{B}_x^2+\mathrm{B}_y^2+\mathrm{B}_z^2-\mathrm{B}_w)-(\mathrm{E}_x^2+\mathrm{E}_y^2+\mathrm{E}_z^2-\mathrm{E}_w) \\ \mathrm{C}_x-\mathrm{E}_x & \mathrm{C}_y-\mathrm{E}_y & \mathrm{C}_z-\mathrm{E}_z & (\mathrm{C}_x^2+\mathrm{C}_y^2+\mathrm{C}_z^2-\mathrm{C}_w)-(\mathrm{E}_x^2+\mathrm{E}_y^2+\mathrm{E}_z^2-\mathrm{E}_w) \\ \mathrm{D}_x-\mathrm{E}_x & \mathrm{D}_y-\mathrm{E}_y & \mathrm{D}_z-\mathrm{E}_z & (\mathrm{D}_x^2+\mathrm{D}_y^2+\mathrm{D}_z^2-\mathrm{D}_w)-(\mathrm{E}_x^2+\mathrm{E}_y^2+\mathrm{E}_z^2-\mathrm{E}_w) \end{vmatrix} \tag{2.12}$$

If all four weights are equal, elementary linear algebra shows that we fall back on the standard unweighted orient3D predicate.

The expressions (2.5) and (2.6), (2.7) and (2.8), and (2.11) and (2.12) are pairwise equivalent and require approximately the same amount of computation. Generally, the second forms — obtained by algebraic operations that map to the translation of the origin on one of the points — are less prone to numerical problems. Nevertheless, if standard floating-point computations are used, special techniques must be used to evaluate correctly the sign of these determinants, see section 4.3.

2.4 Dynamic case

So far we have only addressed the case where the positions $\vec{c}_i$ are fixed. However, for the purpose of our simulations, we need to track the dynamic behavior of the grains and update the set of potential neighbors for which collisions are tested. This essentially means deleting edges from *resp.* adding edges to $\mathcal{DT}$. More precisely, if the centers $\vec{c}_i$ become time-dependent, so will the values of $P_i(\vec{x})$ and over time the minimum for a given $\vec{x}$ can be attained for a different i. This yields a topologically different Laguerre complex and a topologically different weighted Delaunay triangulation, as shown in figure 2.13. Failing to update the adjacency information contained in the triangulation will most likely result in collisions being missed.

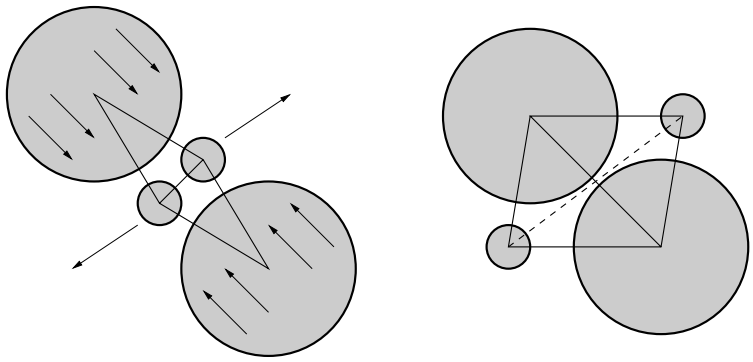


Figure 2.13: As the grains move, the original Delaunay triangulation may fail to detect contacts. Maintaining the triangulation is mandatory: a new edge should appear to reflect the new potential contact, and another one should disappear to maintain the triangulation structure.

2.4.1 Local operations on triangulations

Clearly, recomputing the whole triangulation from scratch is not an option: it takes too much time and the expected benefit of the method vanishes. Fortunately, in dimensions two and three it is possible to transform any triangulation into any other triangulation by applying local operations called *flips*. Performing a flip results in alternating between the two possible triangulations of a set of four points in 2D or of five points in 3D.

Starting from the initial Delaunay triangulation, and given the motion of each center $\vec{c}_i$, the triangulation remains valid as long as the weighted insphere predicate applied to all facets says so. If a portion of the triangulation becomes invalid, a topological change takes place and the triangulation must be updated accordingly. With our strong assumption that no Laguerre cell may become empty and thus never disappear[5], the only operations required are flips.

We now have a criterion to determine where our triangulation differs from the Delaunay triangulation, and local operations to transform it back into such a triangulation. For a given set of

[5]Disappearing Laguerre cells and the corresponding topological operations were used in another context by Xue et al. (1997).

points, the Delaunay triangulation is the only one that maximizes the minimal angle in a triangle. Performing a flip wherever the insphere criterion is violated increases the minimal angle found in the triangles involved in the flip. These nice results allow us to assert that this procedure of flipping wherever detected by the insphere criterion will always yield the Delaunay triangulation after a finite number of flips. In practice, the number of flips to perform depends on the volatility of the media, but for most practical uses is very low.

2.4.2 In 2D

For didactical purposes, we shall first illustrate the triangulation maintenance in 2D. Looking at the situation in figure 2.9, the incircle predicate applied to the triangle {1,2,3} says that point 4 violates the criterion.

The flip replaces triangles {1,2,3} and {1,3,4} by triangles {1,2,4} and {2,3,4}. If we look at the edges of the triangulation only, then the flip is just the exchange of diagonals in the quadrilateral {1,2,3,4}. Of course, for this operation to be possible, {1,2,3,4} must be convex.

If we consider the quadrilateral {1,2,3,4} as the planar projection of a simplex in 3D, then the flip can also be interpreted as alternating between a view from the top and a view from the bottom of that simplex, see figure 2.14.

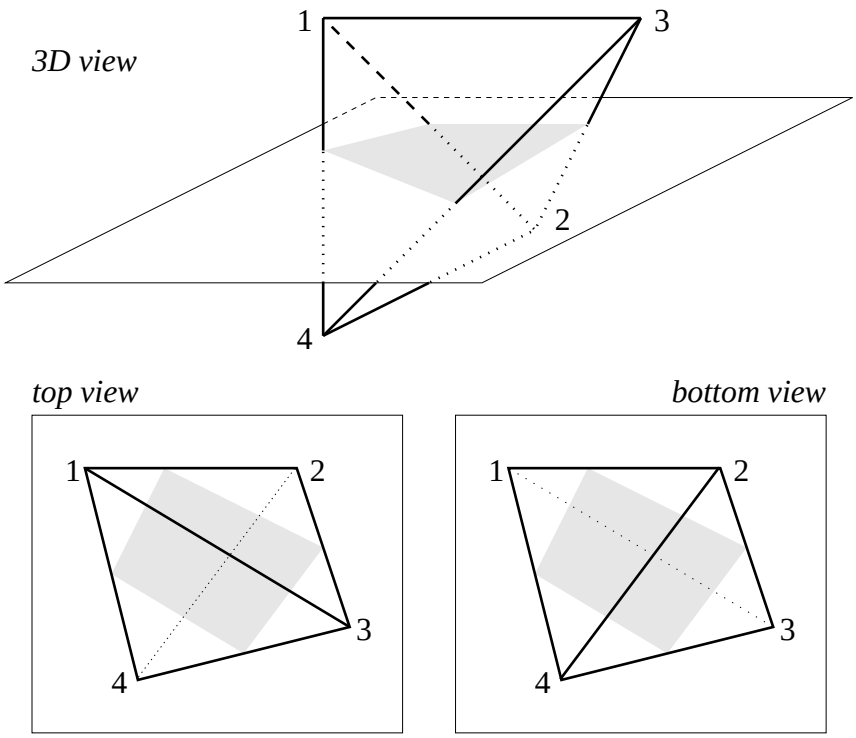


Figure 2.14: The flip in 2D seen as alternating between top and bottom view of a 3D polyhedron. The grayed surface is the intersection with the projection plane, irrelevant in this perspective.

The flip is a symmetric operation, and is the only one required to transform a 2D triangulation.

Furthermore it preserves the number of edges and triangles in a triangulation, which simplifies the implementation.

2.4.3 In 3D

The same principle applies in 3D, but is slightly more complicated because two kinds of flips exist, flip1 and flip2, as shown in figure 2.15.

Flip1 adds an edge between 1 and 5, and replaces tetrahedra {1,2,3,4} and {2,3,4,5} with tetrahedra {1,2,3,5}, {1,3,4,5} and {1,2,4,5}. It is performed when the insphere predicate applied to the tetrahedron {1,2,3,4} says that the point 5 violates the criterion and that facet {2,3,4} should be flipped. Again, {1,2,3,4,5} must be convex for flip1 to be possible.

Flip2 is the inverse operation and removes the edge between 1 and 5, and replaces tetrahedra {1,2,3,5}, {1,3,4,5} and {1,2,4,5} with tetrahedra {1,2,3,4} and {2,3,4,5}. It is performed if the following conditions are met:

- edge {1,5} belongs to exactly three facets (or equivalently three tetrahedra) of the triangulation, and
- the incircle predicate applied to the suitable tetrahedra indicates that all three facets {1,2,5}, {1,3,5} and {1,4,5} should be flipped.

The two configurations shown in figure 2.15 can also be interpreted as the two views of a 5-vertex simplex in 4D projected onto a 3D hyperplane, the projection being looked at from each of the two half-spaces defined by the projection hyperplane.

The limit we impose on the relative sizes of the overlaps guarantees that the topological event causing the addition of an edge happens at a time when the two corresponding grains are still disjoint. So, even if they collide, there is a small time difference between the two events, and $\mathcal{DT}$ is ahead of time. Thus, a reasonably small delay in the update of $\mathcal{DT}$ will not miss any collision. We conclude this section on dynamic triangulations of moving sites by discussing two ways to deal with time and their influence on the simulation of granular media.

2.4.4 Scheduling exact event times

Suppose we start at time t_0 with a correct situation, and all the center trajectories $\vec{c}_i(t)$ are known for some time interval $[t_0, t_0 + h_0]$, where h_0 is the length of the horizon at time t_0. One can compute the exact date $t_1 \in [t_0, t_0 + h_0]$ of the next event. At this stage, we may want to update some of the trajectories to make them valid for a new horizon $[t_1, t_1 + h_1]$, and then compute t_2 and so on.

Two categories of events may occur: the collision of two grains, in which case we update their respective trajectories to reflect an instantaneous shock, or a portion of $\mathcal{DT}$ becoming invalid, leading to performing the suitable flips, the trajectories remaining unchanged.

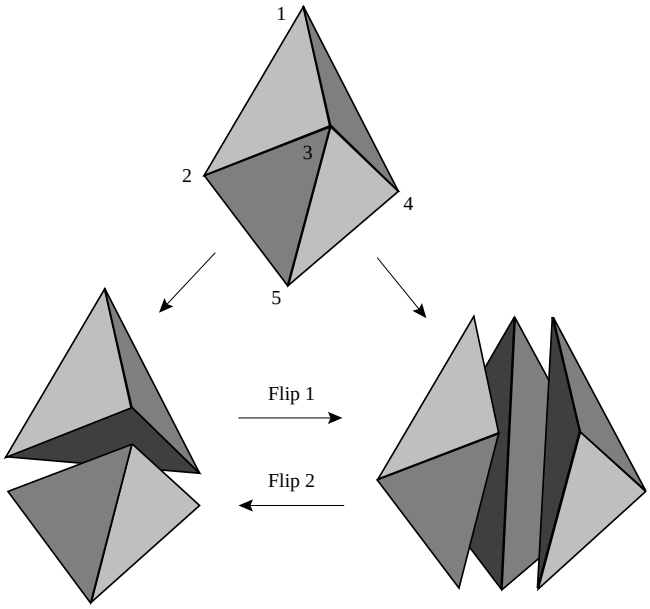


Figure 2.15: The flips in 3D. The set of five points can be triangulated into either two tetrahedra and no internal edge, or three tetrahedra and one internal edge. Flipping is alternating between the two configurations.

Between two successive events, the grains move along known trajectories. For each pair of grains given by the edges (i, j) of the triangulation, the time t_1 of the next collision during time interval $[t_0, t_0 + h_0]$ is the smallest root of

$$\left\| \vec{c}_i(t) - \vec{c}_j(t) \right\| = r_i + r_j \tag{2.13}$$

larger than t_0, if one exists. If the grains are only subject to a constant acceleration field such as gravity, then the above equation is quadratic, trivial to solve. The method remains perfectly valid for movements constrained by non-constant acceleration fields, as discussed by Sigurgeirsson et al. (2000). These happen for example when the grains flow in a viscous fluid.

The detection of the topological events follows similar rules. The determinants of the predicates are built with time dependent coordinates, and finding when they change sign reduces to finding the smallest root of a 5^{th} order polynome larger than t_0. A computationally efficient method for solving this is given in Müller (1996a).

This method is used in 2D granular media simulations based on hard body contact models where the grains never overlap and the trajectories of the colliding grains are updated to reflect their motion after the instantaneous shock, see Müller and Liebling (1995) or Müller (1996a). It is well suited for sparse media: low density, grains moving fast, few collisions... However, these models tend to be inaccurate in static, dense situations because of inelastic collapse due to the inappropriate treatment of the lasting contacts, see McNamara and Young (1992). Furthermore, computing the event dates is very sensitive to rounding errors and scheduling them limits the parallel processing potential of the DEM.

2.4.5 Discretizing time

In this second approach to dealing with it, time is sliced into small intervals δt and events occurring during such an interval are postponed for treatment at the end of the interval. The trajectories $\vec{c}_i(t)$ during $[t_k, t_k + \delta t]$ should be known at t_k and δt should be small enough. This approach means that the triangulation may violate the Delaunay criteria during at most δt, but we have seen that this is acceptable for the collision detection. However, aside from taking the risk of not detecting a collision — or more precisely detecting it too late: at t_{k+1} a collision that started between t_k and t_{k+1} — the risk of ending up with a geometric structure that does not represent a valid triangulation anymore must be considered. This can be the case if a tetrahedron becomes degenerate before a flip occurs (see figure 2.16) or if a grain was completely swallowed by another due to an overlap that grew too large.

This is the method we use in our simulations based on soft body force models since they also discretize time. The choice of the timestep δt is driven by the force model as these models usually require a much higher sampling rate, yielding a "safe" δt for the triangulation maintenance.

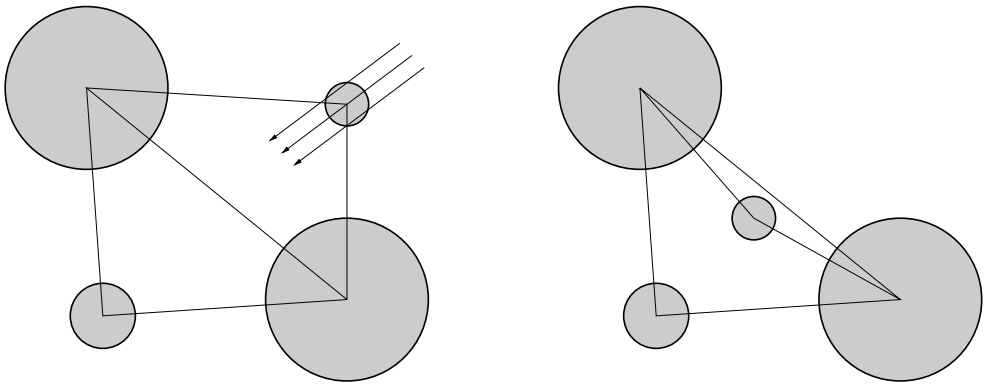


Figure 2.16: The small grain has moved too far between two triangulation updates, the triangulation is not valid anymore. Both the incircle test and the flips are useless to recover from this situation.

2.5 Degenerate cases

So far we have assumed that the centers of the grains were in general position: no three points are collinear, no four points are coplanar and no five points are cospherical. The latter condition is equivalent to saying that no point in E^3 belongs to more than four Laguerre cells, which is also equivalent to saying that the weighted Delaunay triangulation $\mathcal{DT}(\mathcal{K})$ is unique. In 3D, points in general position also guarantee that the number of edges, facets and tetrahedra of $\mathcal{DT}$ is linear, see Boissonnat and Yvinec (1995).

Although the probability of encountering such degenerate configurations is conveniently believed to be zero, we cannot assume that they will never appear. In particular, since we are using standard floating-point arithmetic for our computations, the coordinates of the grains are constrained to a discrete subset of $\mathbb{R}$. Therefore, the chances that some of them line up are higher. However, we will show that those configurations do not affect the validity of the triangulation-based neighborhood, and that they do not cause major implementation difficulties.

2.5.1 Non-uniqueness of the Delaunay triangulation

The non-uniqueness of the Delaunay triangulation does not constitute a threat to the collision detection scheme since pairs of grains associated with the non-mandatory edges will only collide if very large overlaps are allowed. Consider four grains of equal size in 2D such that their centers make a perfect square. The corresponding Delaunay triangulation has as its edges the four edges of the square and either one of its diagonals. Selecting one or the other does not miss a collision, unless the overlap amounts to $(2-\sqrt{2}) \approx 0.58$ times the radius (see figure 2.17) which is explicitly forbidden in our case. Furthermore, since grains move, if either pair of opposite grains gets closer, the degeneracy will be removed. That property extends to the weighted as well as the 3D case. From a practical point of view, if we encounter such a degenerate situation, it will suffice to just pick up one of the possible triangulations. Care must be taken, though, to ensure the consistency of the triangulation. In other words, in the situation mentioned above, any diagonal can be chosen, but exactly one of them must be part of the triangulation. If none or both are taken, the resulting structure is not a valid triangulation anymore.

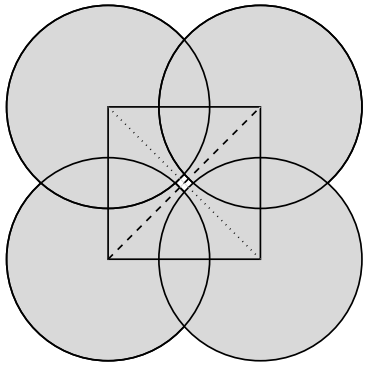


Figure 2.17: If the grains are not in general position, the Delaunay triangulation is not uniquely defined, but no collision is missed. Overlaps larger than shown here would lead to such a situation, but they are explicitly forbidden in this context.

2.5.2 Delaunay triangulation with superlinear number of edges

If the centers are temporarily in a configuration that yields a triangulation with a quadratic number of tetrahedra, triangles and edges, the method will again not fail, but waste a non-negligible amount of time generating the triangulation, detecting contacts, and at some point reducing the triangulation to a linear number of elements once the configuration will no longer be degenerate. However, the configurations for which the only possible triangulations all have a quadratic number of elements are very unlikely in the context of granular media simulations, see such an example in figure 2.18. Should such cases occur too frequently in some situations, the techniques of Edelsbrunner and Mücke (1990) or of Alliez et al. (1998) could be used to lift the degeneracies. Configurations for which there are several possible triangulations, among which some quadratic, are more common. In those cases, the Delaunay triangulation generally minimizes the number of elements (Fukuda, 2001).

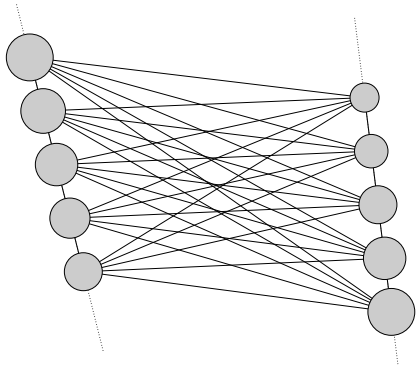


Figure 2.18: A configuration that yields a triangulation with a quadratic number of edges. There are n points lined up on each one of two non-coplanar lines: the only possible triangulation contains $n^2 + 2(n-1)$ edges. If a new point is added in the interior of the convex hull, or if one of the existing $2n$ is slightly shifted *within* the hull, then the Delaunay triangulation will have $O(n)$ edges.

2.6 Conclusion

The validity and efficiency of our approach to collision detection among spheres is established, at least from a theoretical point of view. We are now ready to build a complete DEM algorithm around the three-dimensional weighted Delaunay triangulation, which we will do in the next chapter.

As far as the laws of mathematics refer to reality, they are not certain,
and as far as they are certain, they do not refer to reality.
– Albert Einstein

Chapter 3

The Distinct Element Method

3.1 Introduction

The Distinct Element Method (DEM) is an approach to numerical simulation where statistical measures of the global behavior of a phenomenon are computed from the individual motion and mutual interactions of a large population of elements. It is commonly used in situations where state-of-the-art theoretical knowledge has not yet provided complete understanding and mathematical equations to model the physical system. The major idea behind DEM is to circumvent the complexity of a large assembly by considering instead many simple elements, the behavior of which can be simulated accurately. Because of this approach, DEM requires careful calibration and validation with real experiments in order to produce trustworthy results.

Many other simulation methods exist. When the suitable equations are known to hold for the system, for example the century-old Navier-Stokes equations for fluid flows, numerical simulation can be reduced to the numerical integration of said equations given suitable boundary and initial conditions, most often a very difficult problem. The choice of a simulation method for a given problem is influenced by factors such as the level of theoretical knowledge on which the model can rely, the nature of the application, or the computational resources available. The DEM does not rank very high on the specialists' wish list because, as we just mentioned, it requires careful calibration and validation but also because it usually requires huge number of elements in order to produce realistic results. Such large-scale simulations in turn are very demanding in terms of raw computational power, but also in terms of memory, online storage, pre- and post-processing, etc.

Due to their highly discontinuous nature, one should expect that granular media require a discontinuous simulation method. Indeed, to date DEM is the leading approach to those problems. Modeling is straightforward: the grains are the elements, they interact through local, pairwise contacts, yet are also subject to external factors such as gravitation or contacts with surrounding objects, and they otherwise obey Newton's laws of motion. This is enough for *dry* granular media where grains of any size, shape and material interact. *Wet* granular media where the influence of an interstitial fluid (liquid or gas) cannot be neglected require more attention. Simple attempts have been made to integrate this influence in the numerical contact model or the

equation of motion (see §3.4.1), but to achieve a sufficient degree of realism, the tight coupling of DEM for the grains with the relevant flow simulation may be necessary. That is the price to pay for simulations of wet sand piles, landslides or snow avalanches. Some preliminary results in this direction are available, see Muguruma et al. (2000) and Sigurgeirsson et al. (2000).

The DEM approach is very similar to — and in fact mostly inspired from — Molecular Dynamics (MD). One of the major differences between the two is the driving force: in MD, particles follow the gradient of a global potential, which in turn depends on the relative positions of the particles, while DEM particles interact through local pairwise contacts. Another difference is induced by scale: while MD usually deals with particles at the atomic level and time ranges of pico- or nanoseconds, the sizes in DEM range between macroscopic (rocks) and mesoscopic (powders) and durations of the order of a few seconds. This scale difference in turn induces specialized practices: MD often use periodic boundary conditions to get rid of surface effects and artificially increase the size of the population, while most DEM simulations deal explicitly with a given geometry.

Even if we restrict our scope to DEM simulations of granular media, we can find several different algorithms in the recent literature. The major criteria to distinguish them include the shape of the grains, the modeling of the contacts, the way to deal with time, the method to detect collisions. We have restricted ourselves in this thesis to spherical grains in 3D, with soft body contact models and discretized time. Chapter 2 presented the core element of our approach, the efficient detection of collisions using a dynamic triangulation, this chapter will now build a complete simulation method around it. Again, most implementation details are postponed to the next chapter.

3.2 The DEM algorithm for spherical grains

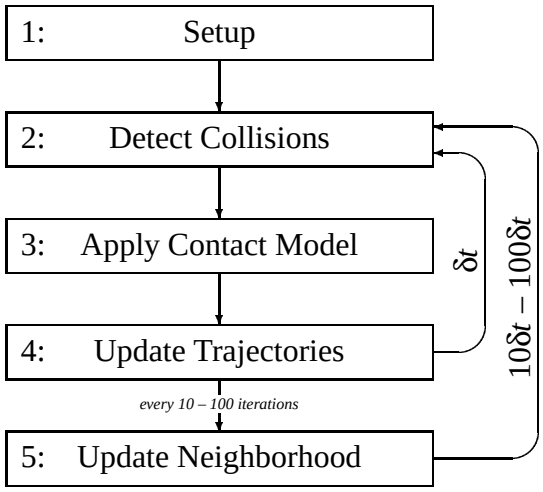


Figure 3.1: The DEM algorithm for spherical grains.

Figure 3.1 gives a schematic representation of the algorithm. We describe now each step in detail:

1. **Setup**
   Setting up the simulation consists in generating an initial set of n grains and arranging them in the medium (although more grains can be added later), defining boundaries for the container, and initializing any internal structure used during the simulation. In our case, the initial triangulation is built incrementally as the grains are generated, see section 4.2.1.

2. **Detect collisions among the grains**
   Locate pairs of colliding grains. To avoid performing this detection among all $n(n-1)/2$ potential pairs, some sort of neighborhood information is used. This is where the dynamic triangulation shows its strength reducing the complexity of this step from $O(n^2)$ to $O(n)$.

3. **Apply the appropriate contact model at every collision**
   Once all collisions have been identified, proper action must be taken according to the nature of the simulation. This is achieved by applying a numerical contact model based on values like the size of the overlap, relative normal and tangential speeds, particle types, etc.

4. **Update the trajectories of the grains**
   After summing the forces contributed by the contacts in step 3 and external forces (gravitation, contact with the boundaries, etc), it is now possible to integrate the motion equations over the duration of the iteration, resulting in new particle velocities, positions, spin and orientation.

5. **Update neighborhood**
   Advance clock by δt and shift grains to their new positions. However, since the grains move, the neighborhood information used in step 2 remains valid only for a limited period and has to be updated every few iterations.

6. **Go back to step 2**

The setup is not critical. Some implementation details will be given in chapter 4, and each experiment in chapters 5 and 6 will describe what kind of grains, boundaries and other conditions are used. Steps 2 and 5 were addressed in chapter 2. Sections 3.3 and 3.4 describe the remaining two steps of the algorithm.

The iteration duration δt is dictated by the computational model used to represent the contact in step 3 and the numerical integration of Newton's equations of motion in step 4. The soft body theory assumes a contact duration of about 10^{-5} seconds and requires at least 100 iterations per contact to guarantee numerical stability, thus setting $\delta t = 10^{-7}$. This is too small for today's computers, though, and most authors admit using a much larger value, in the 10^{-4} to 10^{-5} range. This value is also acceptable for step 4. How often the neighborhood must be updated in step 5 depends on the neighborhood technique used, but also on the volatility of the media. Usually this happens only every 10 to 100 iterations, which reduces the impact of this step on the overall computation time.

3.3 The contact models

In the DEM algorithm outlined in section 3.2, step 3 relies on a numerical contact model for the proper simulation of the collision between two grains. The full description of the physical phenomena involved in such an interaction and the mathematical tools used to accurately replicate them is an ongoing effort of many researchers. We have used some of the widely accepted models, but have also been able to test the latest development of Pournin et al. (2001). We now present the steps required to go from the geometric event represented by the collision of two spheres to the physical force exerted at the contact point. We are using soft body models that compute a force from an overlap and we discretize time. At each iteration the force model is responsible to update the force to reflect the new situation. Figure 3.2 shows the contact between two grains and the various quantities used.

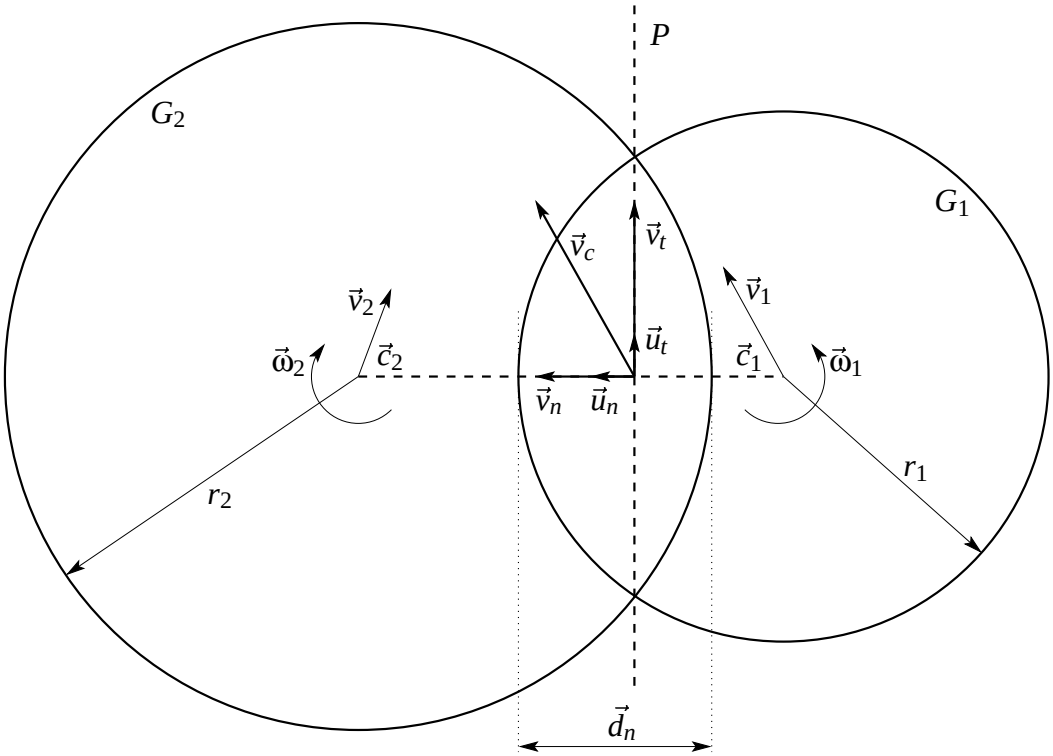


Figure 3.2: A schematic view of a contact between two grains.

Grains G_1 and G_2 collide at time[1] t when the distance $\|\vec{c}_1 - \vec{c}_2\|$ between their centers is smaller[2] than the sum of their radii $r_1 + r_2$. The size of the overlap

$$d_n = r_1 + r_2 - \|\vec{c}_1 - \vec{c}_2\| \tag{3.1}$$

is extremely exaggerated in figure 3.2, as it is in general limited to a few percent of the grain sizes. From now on we consider the collision of grain G_2 on grain G_1, thus generating a force $\vec{F}_{12}$ on G_1. Elementary physics tells us that a force $\vec{F}_{21} = -\vec{F}_{12}$ is then exerted on G_2 by the

[1] For the clarity's sake, we have omitted the time dependence in some equations. All quantities except the radii r_i are in fact time-dependent.

[2] The soft body models do not generate any force when the grains touch each other without overlap.

same contact, the ordering of the grains serves only to orient the various vectors used in the discussion. Let

$$\vec{u}_n = \frac{\vec{c}_2 - \vec{c}_1}{\|\vec{c}_2 - \vec{c}_1\|} \tag{3.2}$$

be a unit normal vector for the contact and P the contact plane normal to $\vec{u}_n$. The relative velocity at the contact point is given by

$$\vec{v}_c = \dot{\vec{c}}_2 - \dot{\vec{c}}_1 + \vec{u}_n \times (r_2\vec{\omega}_2 + r_1\vec{\omega}_1) \tag{3.3}$$

We can project $\vec{v}_c$ on $\vec{u}_n$, thus obtaining the normal part of the relative velocity

$$\vec{v}_n = (\vec{v}_c\vec{u}_n)\vec{u}_n \tag{3.4}$$

Note that

$$\|\vec{v}_n\| = \dot{d}_n \tag{3.5}$$

as one would expect.

We can also project $\vec{v}_c$ on the plane P, thus obtaining the tangential part of the velocity

$$\vec{v}_t = \vec{v}_c - \vec{v}_n = \vec{v}_c - (\vec{v}_c\vec{u}_n)\vec{u}_n \tag{3.6}$$

If $\vec{v}_t = \vec{0}$ then the collision is purely normal, the force $\vec{F}$ will be collinear with $\vec{u}_n$ and no torque will be generated. If $\vec{v}_t \neq \vec{0}$, the unit tangential vector

$$\vec{u}_t = \frac{\vec{v}_t}{\|\vec{v}_t\|} \tag{3.7}$$

can be defined but is usually not required. $\vec{v}_t$ is the instantaneous tangential relative velocity at the contact point. Integrating this value over the duration of the contact gives the distance covered by the initial contact points along the surface of the grains, a value used by some contact models:

$$\vec{d}_t = \int \vec{v}_t \mathrm{dt} \tag{3.8}$$

3.3.1 Simple contacts

The above discussion is sufficient to introduce a first class of contact models that deal with isolated contacts. Such contacts are divided in two phases, a *loading*, when $\dot{d}_n > 0$ followed by an unloading, when $\dot{d}_n < 0$. A soft body model must provide a pair of functions F_n and $\vec{F}_t$ that will compute a normal and a tangential component for the repulsion force from the normal and tangential overlap and their first time derivative:

$$\vec{F} = F^n(d_n, \dot{d}_n)\vec{u}_n + \vec{F}^t(\vec{d}_t, \dot{\vec{d}}_t) \tag{3.9}$$

Note that the normal component is one-dimensional, as the normal force always follows the normal vector of the contact, while the tangential component is a $(d-1)$-dimensional vector — a two-dimensional vector for 3D simulations — as the tangential force always follows the contact plane P.

In fact, equation (3.9) must be slightly modified because Coulomb's law of friction limits the tangential force

$$\|F^t\| \leq \mu \|F^n\| \tag{3.10}$$

where μ is the friction coefficient.

Much more effort has been devoted to normal forces than to their tangential counterparts in the design of models and in the fitting of the parameters. In fact, some contact models give elaborate expressions for the normal part and fall back on traditional linear expressions for the tangential part. The reason for this is mostly due to the fact that one of the most critical effects of numerical contact models, the dissipation of energy, is almost entirely carried by the normal force. However, tangential forces play a central role in some phenomena such as convection and interaction with boundary objects.

One of the simplest contact models, first introduced by Cundall and Strack (1979), mimics a double linear spring and dashpot system, as shown in figure 3.3. The normal and tangential components F^n and $\vec{F}^t$ of the force are obtained by adding an elastic and a viscous contribution, both of them computed as linear functions of the overlap size and relative velocity:

$$F^n = k_n d_n + c_n \dot{d}_n \tag{3.11}$$

$$\vec{F}^t = k_t \vec{d}_t + c_t \dot{\vec{d}}_t \tag{3.12}$$

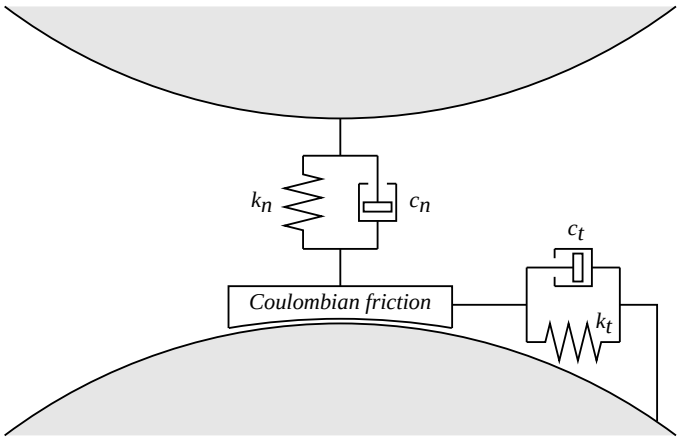


Figure 3.3: Schematic representation of Cundall's model for the contact between two grains.

In the original model by Cundall, the elastic and viscous coefficients k_n and c_n introduced as parameters of the simulation cannot be matched to physically measurable or interpretable quantities. Choosing realistic values for them is a problematic task, especially with regard to energy dissipation. It is possible, though, to map the simulated behavior with available experimental data for a single collision between two grains, thus providing expressions for k_n and c_n as function of the duration of the contact t_c and the energy dissipation ratio e_n:

$$k_n = \frac{m_{\text{eff}}}{t_c^2}(\pi^2 + \ln(e_n)^2) \tag{3.13}$$

$$c_n = -\frac{2m_{\text{eff}}}{t_c}\ln(e_n) \tag{3.14}$$

where $m_{\text{eff}} = \left(\frac{1}{m_i} + \frac{1}{m_j}\right)^{-1}$ is the *equivalent mass* for the two grains i and j of mass m_i and m_j. Cundall and Strack themselves suggested choosing k_t and c_t equal to their normal counterparts.

3.3.2 Multiple contacts

This is not sufficient yet: calibrating the parameters using experimental data for a single collision does not provide accurate energy dissipation whenever a particle is involved in several contacts at the same time. This is due to the fact that such contacts do not follow the same loading - unloading scheme as isolated contacts, but generally involve several reloading phases. Purely viscoelastic models fail to provide accurate energy dissipation during the reloading phases. Elastoplastic models such as the one introduced by Walton and Braun (1986) improve this dissipation, but have other problems of their own such as loss of control on the maximal size of the overlaps. Recently, Pournin et al. (2001) proposed a new approach designed to allow full control on the energy dissipated during the successive phases of a contact. To achieve this, the behavior of every new unloading phase is set according to the amount of energy dissipated so far and the total amount of energy to be dissipated by the whole contact. If further reloading phases occur, a new marginal dissipation will occur during the next unloading phase, as shown in figure 3.4.

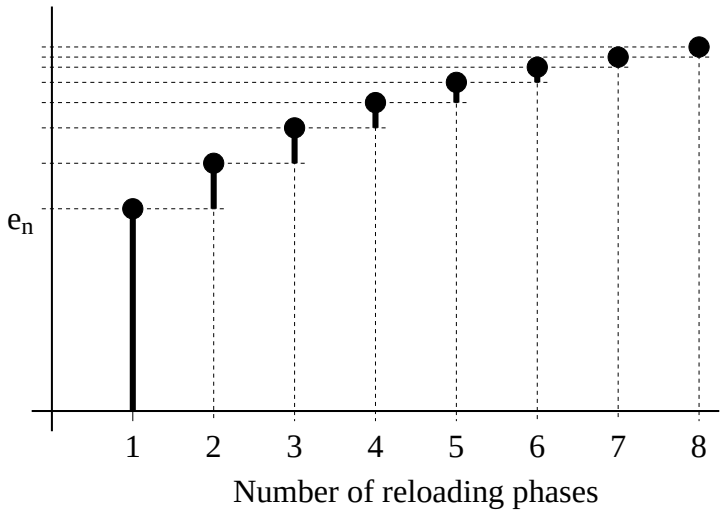


Figure 3.4: Adaptive energy dissipation as proposed by Pournin et al. The large dots give an experimental approximation of the energy dissipation in an n-reloading contact, the numerical models dissipate at each reloading phase the amount given by the thick line.

Furthermore, the individual control of each reloading phase lifts the strong hypothesis made by other models that contacts involving the same grain are independent. Since energy dissipation at every new reloading phase is re-evaluated, the hysteresis of the contacts can be integrated in each new phase.

To understand how Pournin's new model works, it helps to look at the force-overlap diagrams for various contact models, see figure 3.5. A purely elastic force model dissipates no energy.

Cundall's model introduces energy dissipation through the viscous component, yielding the elliptic trajectory on the diagram. The grayed area, the integral of this trajectory, is the amount of energy dissipated. Walton's model accounts for a plastic deformation, but the reloadings follow the same trajectory and thus fail to dissipate more energy, unless the reloading occurs relatively late (dark gray area). Pournin's model chooses independent slopes for each loading and unloading phase, yet controls the total deformation by the non-plastic return phase.

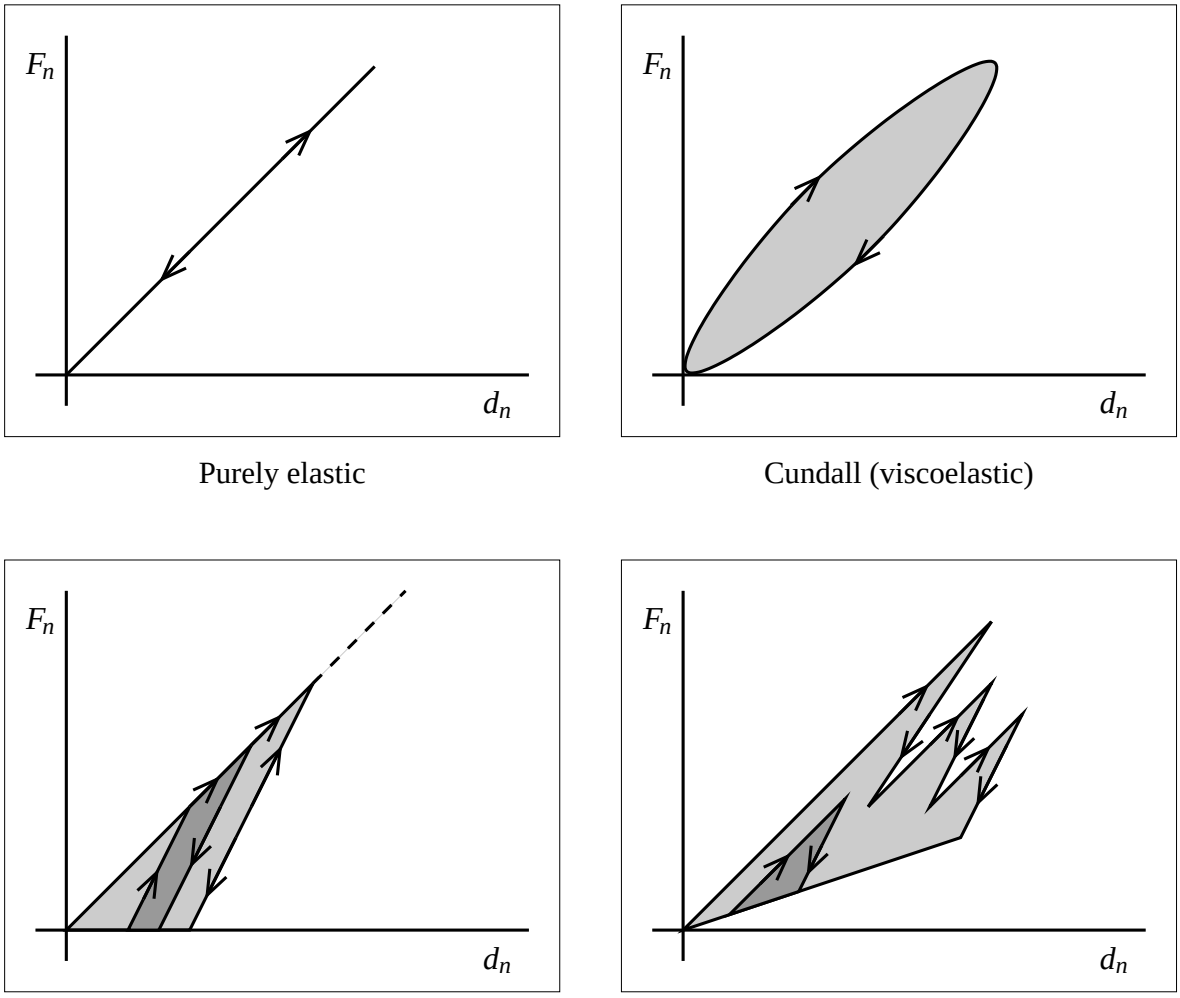


Figure 3.5: Force-overlap diagrams for some contact models. The arrows represent the various loading and unloading phases. The gray part represents the energy dissipated by the contact.

3.3.3 Contacts with walls

The theory developed for the contacts between two grains may be applied to the contact of a grain with a wall. It suffices to assimilate the wall with a grain of infinite radius. In particular, it means that the vector $\vec{u}_n$ is perpendicular to the wall at the contact point, the plane P is the

tangential plane to the wall at the contact point. Planar walls as well as cylindrical, conical and spherical walls have been used in our simulations.

Fixed walls can be seen as having an infinite mass and are thus motionless, but mobile walls such as the piston used to compress grains in some of the experiments described in chapter 5 will have a mass of their own and their behavior will also depend on the forces exerted on them by the grains.

3.4 Integrating the motion equations

In step 4 of the algorithm we assume that all the forces acting on a grain are known. These forces come from contacts with other grains — as computed in step 3 and discussed in the previous section — or from contacts with the walls or floor, or from external factors like gravitation. We can thus compute the total force acting on grain i at time t

$$\vec{F}_i(t) = \sum_{(i,j)\in\mathcal{DT}} \vec{F}_{ij}(t) \; + \; \vec{F}_{i\,\text{walls}}(t) \; + \; \vec{F}_{i\,\text{gravity}}(t) \tag{3.15}$$

Note that we take the sum over all edges of $\mathcal{DT}$ incident to grain i, yet there may be edges along which no contact occur. Those edges will have to return no force. Similarly, the tangential components of these forces, computed separately by most numerical contact models and limited by Coulomb's law of friction, generate a torque

$$\vec{T}_i(t) = \sum_{(i,j)\in\mathcal{DT}} r_i\vec{u}_n \times \vec{F^t}_{ij}(t) \; - \; r_i\vec{n} \times \vec{F^t}_{i\,\text{walls}}(t) \tag{3.16}$$

where $\vec{u}_n$ comes from equation (3.2) and $\vec{n}$ is a unit vector normal to the wall at the contact point directed towards the grain, which explains the minus sign. We can then use Newton's equations of motion to translate and rotate each grain:

$$m_i\ddot{\vec{c}}_i(t) \;=\; \vec{F}_i(t) \tag{3.17}$$
$$I_i\dot{\vec{\omega}}_i(t) \;=\; \vec{T}_i(t) \tag{3.18}$$

where I_i is the moment of inertia of grain i, with $I_i = \frac{2}{5}m_i r_i^2$ if the grains are homogenous.

Numerical integration of these equations goes as follows. The sum of the forces $\vec{F}_i$ yields an acceleration $\vec{a}_i$ at time t for grain i

$$\vec{a}_i(t) = \frac{1}{m_i}\vec{F}_i(t) \tag{3.19}$$

From the current positions and velocities $\vec{c}_i(t)$ and $\dot{\vec{c}}_i(t)$ we can then predict new positions $\vec{x}_i$ and velocities $\vec{v}_i$ at time $t+\delta t$:

$$\vec{v}_i(t+\delta t) \;=\; \dot{\vec{c}}_i(t) + \delta t\,\vec{a}_i(t) \tag{3.20}$$
$$\vec{x}_i(t+\delta t) \;=\; \vec{c}_i(t) + \delta t\,\dot{\vec{c}}_i(t) + \frac{1}{2}\delta t^2\,\vec{a}_i(t) \tag{3.21}$$

To obtain sufficient precision for long time steps, *predictor-corrector* methods use now $\vec{x}_i(t+\delta t)$ and $\vec{v}_i(t+\delta t)$ to compute the forces and accelerations at time $t+\delta t$, and use this information to

correct the predicted values $\vec{x}_i(t+\delta t)$ resp. $\vec{v}_i(t+\delta t)$ into $\vec{c}_i(t+\delta t)$ resp. $\dot{\vec{c}}_i(t+\delta t)$, then iterate from there. However, as the numerical contact models we use impose very short time steps, this correction step is usually not required.

Likewise, the torque $\vec{T}_i$ generates an angular acceleration $\vec{\alpha}_i$ at time t for grain i

$$\vec{\alpha}_i(t) = \frac{1}{I_i}\vec{T}_i(t) \tag{3.22}$$

Angular velocity and positions are then computed like their directional counterparts.

3.4.1 Linear damping

Cundall and Strack (1979) introduced a linear approximation for the friction caused by the interstitial fluid in the form of directional and rotational damping. The standard equations of motion are modified and become

$$m_i\ddot{\vec{c}}_i + C\dot{\vec{c}}_i = \vec{F}_i \tag{3.23}$$
$$I_i\dot{\vec{\omega}}_i + C^*\vec{\omega}_i = \vec{T}_i \tag{3.24}$$

where C and C^* are the directional and rotational damping coefficients. This is sufficient for the cases where the friction is limited, but does not provide feedback from the grains to the fluid, a key issue for realistic coupling of DEM and fluid models. Due to the dry nature of our simulations we do not use them at all.

This damping effect dissipates energy, albeit not at the level of the contact between the grains but at the level of the grains trajectories. However, the net effect on the granular assembly is indeed increased energy dissipation, and its use somehow conceals the lack of dissipation of the contact model itself.

3.5 Beyond spherical grains

By looking at spherical grains, we have not only been able to use our triangulation-based collision detection, we have also simplified the actual treatment of the collisions on the grains. However, as Abraham Maslow said, *"When the only tool you own is a hammer, every problem begins to resemble a nail"*. If a mathematical model based on spherical grains is sufficient for a first approach of many phenomena, if many granular materials are indeed composed of "almost" spherical grains, other cases require different grain shape to obtain realistic results. In the remaining part of this chapter we will identify some of the difficulties that prevented so far the extension of the polygonal model of Müller and Liebling (1995) to three dimensional polyhedral grains, then we present an extension of our spherical model in the direction of more general grain shapes.

Müller's idea in 2D works as follows (see figure 1.3 on page 8): A triangulation is built on the vertices of the grains and constrained by the edges of the grains, resulting in a decomposition

of the empty space between the grains. Any movement - translation, rotation or both - of the grains will thus be transmitted to the triangulation, possibly causing a *degeneracy*, that is the disappearing of a triangle due to one of its vertices reaching the opposite edge.

- If that edge belongs to a grain, then there has been a contact, and an appropriate hard body, instantaneous collision model is used to update the trajectories of those two grains.
- If that edge does not belong to a grain, a flip is performed to guarantee the triangulation remains valid.

Given the positions and trajectories of all the grains, therefore of all vertices, it is possible to compute exactly for every triangle when it will become degenerate. All these events are inserted in a scheduler. The simulation then loops forever, taking care of the next degeneracy, updating the triangulation and adjusting future events.

That technique cannot be used in 3D as it is, because a 3D polyhedral shape may not admit a decomposition in tetrahedra. The smallest and simplest example of a non-triangulable polyhedron was already given in 1928 by Schönhart (reported, among others, by Boissonnat and Yvinec, 1995) and is shown in figure 3.6. This means, if at any time the position of the grains is such that some part of the space between them cannot be triangulated, the whole detection process fails.

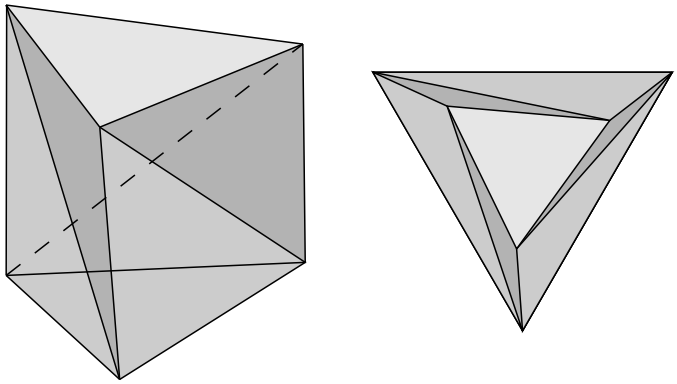


Figure 3.6: Schönhart's polyhedron, 3D view (left) and deformed top view (right). It is basically a twisted, triangle-based cylinder.

Here are two ways to possibly get around this limitation and still apply this method in 3D, both of which are still under theoretical study:

1. **Restrict the shapes of the grains**
   Even the restriction of the grains to regular tetrahedra of the same size may not be enough to guarantee the existence of a triangulation between them. Furthermore, such a restricted model would not be a big improvement over spheres. Using non-regular tetrahedra certainly leads to potentially non-triangulable area such as Schönhart's polyhedron.
2. **Use additional points**
   By adding enough auxiliary points (Steiner points), any space bounded by piecewise linear surfaces can be triangulated. The problem here is to balance between adding enough

points to succeed, and not adding too many (or even remove some when they are not needed anymore) in order to remain efficient, especially in a dynamic context.

It is not clear whether that detection scheme will ever work in 3D. If it does, it will probably require as in 2D a hard body contact model. Work on soft body models for 3D polyhedral grains has already started, see Matuttis et al. (2000), but their dependency on overlaps might well defeat this approach to collision detection.

In the mean time, we tried to build grains of various shape by gluing together spheres into *clusters*, as shown in figure 3.7. This is achieved by replacing, between pairs of spheres that belong to the same cluster, the traditional repulsive force by another force model responsible of keeping them together. From the point of view of the triangulation nothing special must be done. External edges that link spheres from different clusters will support traditional contact models, while internal edges that link spheres within the same cluster will support the gluing force model. Since we consider only convex clusters, internal edges will never disappear, while external edges will come and go to reflect the necessary flips to maintain the Delaunay triangulation. This simply means that we now have a *constrained* Delaunay triangulation. The set of constraints on the triangulation is given by the internal structures of the clusters, and somewhat limited by their relative position. In particular, if we forget the spheres and look at the polyhedra defined by the triangulation within each cluster, we see that those polyhedra will never collide.

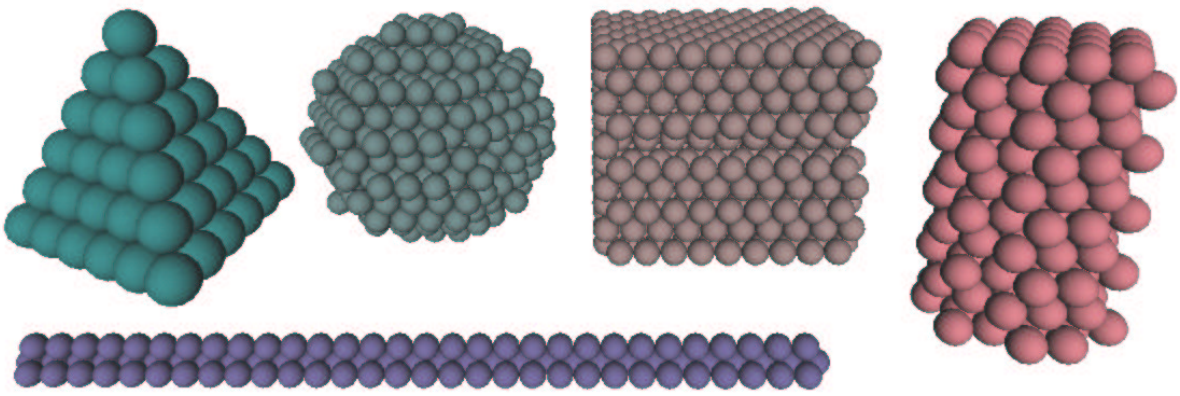


Figure 3.7: Some non-spherical grains obtained by gluing together spheres.

The internal gluing force model still produces a repulsive force when the grains overlap, but also produces an attractive force when the grains are separated. Two such forces have been implemented and tested.

The first force is a simple power of the distance from the equilibrium:

$$F = A\left(\|\vec{c}_i - \vec{c}_j\| - r_i - r_j\right)^{\alpha} \tag{3.25}$$

This expression is purely artificial. It was not derived from any physical property but was chosen because it gave a very rigid behavior to the clusters. Some simulations involving tetrahedra and short rods built with this cohesion force are presented in section 6.4. This first approach allowed us to validate the concept of clusters.

And indeed, validation was necessary. Recalling the comments of section 2.5 about non-degeneracy, we see that these clusters run into two potential problems: the way they are built by placing spheres of equal radii in a face-centered cubic packing means that within a cluster the centers are often collinear. And whenever two long rods are not parallel with nothing in between, we create exactly the situation of figure 2.18 (on page 30). In such cases, if m_1 and m_2 are the lenths of the rods, there will be some $O(m_1 m_2)$ edges between them. Theory warns us that there might be difficulties, and practice has shown that our implementation of the triangulation could very well manage such delicate situations.

Once this approach was validated, we went on to provide a more physically realistic force model between the grains, inspired from the behavior of the atoms in a molecule. The widely used potential of Lennard-Jones is given by:

$$V = Ax^{-\alpha} + Bx^{-\beta} \qquad \text{with } \alpha, \beta > 0 \tag{3.26}$$

where $x = \|\vec{c}_i - \vec{c}_j\|$ is the distance between the centers and A, B, α and β are parameters. The general shape of such potentials is shown in figure 3.8.

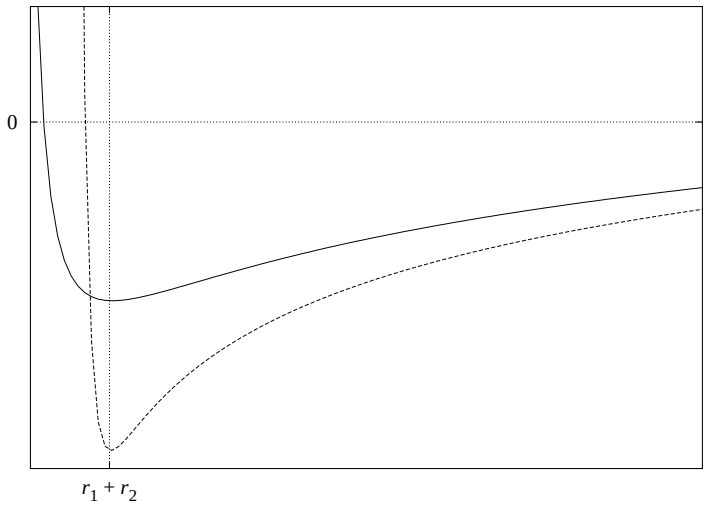


Figure 3.8: Two examples of cohesion potentials.

Deriving this potential yields a force:

$$F = -\alpha A x^{-\alpha-1} - \beta B x^{-\beta-1} \tag{3.27}$$

The problem of assigning pertinent values to the four parameters shows up again. There is a condition of equilibrium when the distance between the centers is exactly $r_1 + r_2$. This must correspond to the minimum value of the potential V and to a force F equal to zero. This condition gives us a first constraint for the set of parameters.

$$\alpha A (r_1 + r_2)^{-\alpha-1} + \beta B (r_1 + r_2)^{-\beta-1} = 0 \tag{3.28}$$

We can use this relation to express B as function of A, α and β:

$$B = -\frac{\alpha}{\beta} A (r_1 + r_2)^{\beta-\alpha} \tag{3.29}$$

Section 6.4 presents preliminary experiments intended to map the values of the remaining parameters to physical quantities describing the material such as Young's modulus.

Computers are useless. They can only give you answers.
– Pablo Picasso

Chapter 4

Computational aspects

The dynamic triangulation presented in chapter 2 used as collision detection mechanism in the framework of the DEM algorithm presented in chapter 3 offers new perspectives in terms of applications. However, the computational bridge between theory and practice remains to be built. Aside from the traditional implementation issues of the DEM, the dynamic triangulation requires extra attention in two crucial areas of computational geometry: efficient data structures for storing and manipulating the triangulation and robustness of the computation with respect to rounding errors. We will first discuss the overall architecture of the simulation environment we have developed, and then address these two major issues. The particularities of the parallel code are then presented, and we conclude with some remarks about other features of our implementation.

4.1 Design objectives

Several modules are involved in the simulation environment. A geometric module provides an implementation for three-dimensional dynamic weighted Delaunay triangulations. It deals with weighted points, segments, triangles and tetrahedra in 3D and provides core functionalities such as insertion of new points and maintenance of the triangulation with respect to the motion of the points.

This geometric module supports the DEM module. The latter deals with physical grains, numerical contact models, and physical boundaries.

The geometric module can be used for other purposes. An attempt was made to reconstruct full dynamic meshes for efficient visualization of SPH[1] simulations but was not pursued. Another example would be to maintain a dynamic mesh for numerical integration wherever it makes sense to have the integration points follow a given path. It would also be possible to use the DEM module with another underlying method for the collision detection.

[1] Smoothed Particle Hydrodynamics (SPH) is a simulation method in which the equations of a flow are solved on a set of virtual particles, see Monaghan (1992).

The modules also use low level functionalities related to the management of the parameters, the generation of pseudo random numbers, etc. Every class of every module is able to dump and restore itself, thus implementing a checkpointing mechanism useful for error recovery or for slicing computations in case of limited resource availability.

A container module that actually stores most objects provides data management. It is responsible for providing controlled access to every element of the simulation either to other elements requesting information (for example a grain that wants to add all the forces acting on it) or to the pool of exporters that produce statistics or files of various formats.

On top of the geometric and DEM modules, a simulation loop orchestrates the various elements and implements the DEM algorithm of section 3.2. Two such loops exist: one for single-processor computations and one multi-threaded designed specifically for shared memory multi-processor machines.

This structure is shown in figure 4.1.

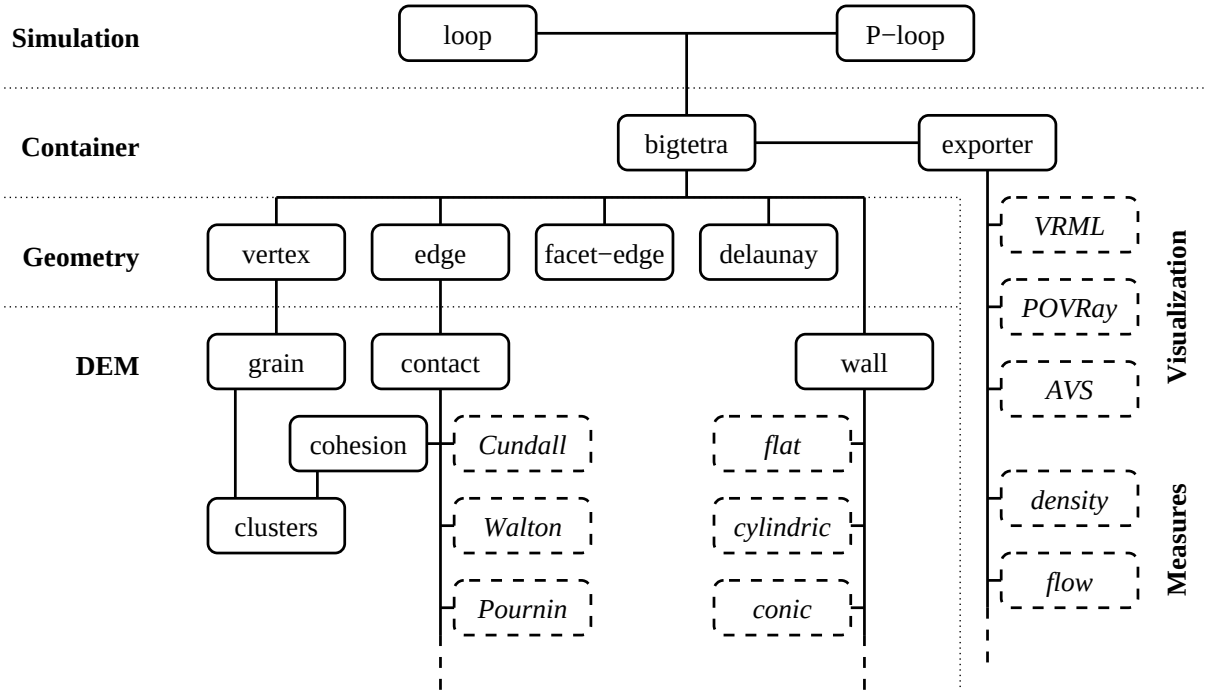


Figure 4.1: The structure of the simulation environment, split in modules and classes.

All the code is written in C++. This choice was driven by the wide availability of the language, the built-in support for intermediate data structures through the Standard Template Library (STL), and the support of the object oriented programming paradigm.

4.2 Implementation of the triangulation

We describe now the implementation of the core of the simulation environment, the triangulation. Most functionalities required for the simulations will be offered through operations on the

triangulation. We shall first recall those operations, and then present two alternative implementations.

4.2.1 Operations required on the triangulation

For the triangulation-based neighborhood to fulfill its role, several functionalities are required: insert a grain, iterate over all potential collisions, add all forces acting on a grain, iterate over all grains. From the perspective of the triangulation, these features map to a set of basic operations to which are added the internal requirements for the triangulation maintenance: iterate over all facets, evaluate the insphere criterion, perform flips wherever they are required.

4.2.1.1 Create an initial triangulation

As mentioned in chapter 1, the construction of a Delaunay triangulation is a well-covered topic. However, our purpose here is not limited to obtaining the triangulation, we need to have access to the triangulation structures in a format that allows easy and efficient updates. Therefore, this initial step does not reduce to producing a mere list of edges, facets and tretrahedra. On the other hand, the performance of this step is not critical, as it is only performed once at the beginning of a simulation, not even when we reload a previous situation to continue the simulation. Furthermore, we wanted to be able to keep adding new grains to the simulation, thus requiring some sort of incremental method.

The scheme we have chosen uses one large tetrahedron enclosing the whole simulation area. Grains are then incrementally added inside this external tetrahedron.

4.2.1.2 Insert a new vertex in the triangulation

The external tetrahedron guarantees that we are always inserting new points inside an existing tetrahedron, which allows us to use the following procedure, illustrated in figure 4.2:

1. Start from any existing tetrahedron of $\mathcal{DT}$.
2. Is the new point inside the current tetrahedron ? If yes, go to 4.
3. Choose a separating facet for the new point from the current tetrahedron among the facets of the latter, move to the adjacent tetrahedron on the other side of this facet, and go back to 2.
4. Split the current tetrahedron into four new ones, thus adding the new point.
5. Optional: enforce the Delaunay property by checking and eventually flipping the facets of the newly added tetrahedron.

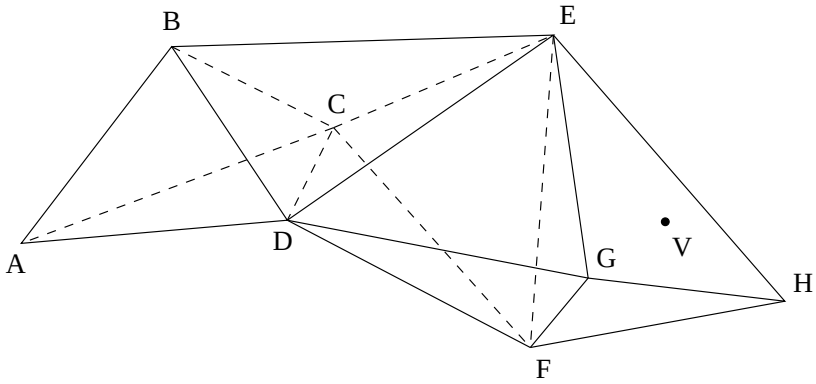


Figure 4.2: Traversing the triangulation to find the tetrahedron in which the point to be inserted lies. Starting from a tetrahedron A,B,C,D, the check on facet B,C,D shows the point V on the bad side of the facet. The adjacent tetrahedron {B,C,D,E} is thus selected. From there the path goes to {C,D,E,F}, {D,E,F,G} and finally to {E,F,G,H}.

The last step is optional: If we do not perform it, we proceed with a perfectly valid triangulation that may not be the Delaunay triangulation. This is intolerable when adding grains during the simulation, but acceptable when building up the initial situation, before any collision detection takes place. The transformation of this incremental triangulation into the Delaunay triangulation is done only once when all grains have been added.

The efficiency of this incremental method for the initial construction of $\mathcal{DT}$ is highly dependant on the choice of the starting tetrahedron in step 1. Guibas et al. (1992) proposed a randomized approach to guarantee a reasonable amortized complexity. As we are usually inserting grains according to a certain pattern, for example on the vertices of a lattice, we can usually choose a starting tetrahedron close to the one containing the new point, thus reducing the number of iterations required to find it.

Step 3 uses the orient3D predicate described in section 2.3. Numerical problems are avoided by using the techniques described below in section 4.3.

The actual implementation of step 4 depends on the underlying data structure, which we will describe later. It consists in adding one new vertex, four new edges, 6 new facets, and updating all the necessary topological information contained therein.

Clearly, if the new point lies exactly on one face of an existing tetrahedron, the insertion procedure will fail, as one of the newly created tetrahedra will be flat. Fortunately, the predicate detects those situations and the application has the choice of either dropping this grain or slightly shifting it in order to remove the degeneracy.

4.2.1.3 Enforce the Delaunay criteria on the triangulation

There are two occasions during the simulation when the triangulation may not be the Delaunay triangulation and thus needs to be updated in order to fulfill the requirements of the collision detection.

- When the grains move, the adjacency relations in the underlying Laguerre complex may change, thus the topology of the Delaunay triangulation changes also. This affects the whole triangulation.

- When one new grain is added, the Delaunay criterion must be enforced locally where the insertion took place.

Both cases follow the same algorithm:

1. Build a list of facets to be checked. This includes either all the *internal* facets of $\mathcal{DT}$, or only those that were involved in the insertion.

2. Apply the weighted *insphere* predicate (see below, section 4.3) to the facets and make a list with the invalid ones.

3. Iterate over that list and perform the flips where they are required and possible, remove the corresponding facets from the list, check the newly created facets and add them to the list if they are invalid.

4. Go back to step 3 until the list is empty.

Unlike in 2D where only one kind of flip is possible, step 3 must decide, for a given invalid facet, which flip to apply. The conditions given in §2.4.3 are used to decide whether an edge will be added or deleted.

Edelsbrunner and Shah (1996) proved that this algorithm works. Work means here that the list of invalid edges will be empty after a limited number of flips, but also that the resulting triangulation is the Delaunay triangulation.

4.2.1.4 Access the vertices of the triangulation

Obviously the vertices of the triangulation, and thus the grains of the simulation, must be stored in a way that allows iterating through them all.

4.2.1.5 Access the edges of the triangulation

Just like the vertices, it must be possible to iterate through all edges of the triangulation, mostly for the efficient computation of the forces but also for exportation purposes. Unlike the vertices, though, this requirement must be accounted for explicitly, as the internal triangulation management may not provide such functionality.

4.2.1.6 Access the edges incident to a vertex

When computing the total force acting on a grain, all edges incident to a vertex must be considered. As this operation is not trivially provided by a 3D triangulation, efficiency suggests that each vertex keeps a list of those incident edges, the list being updated whenever an edge is added or deleted.

One could do without this by simply adding the newly computed force to the corresponding grains, thus effectively combining step 3 and part of step 4. This would be valid in a sequential algorithm where concurrent accesses do not occur, but when we compute the forces in parallel (see section 4.5), this methods creates race conditions that must be explicitly avoided by using some sort of locking mechanism. Since it does not involve any overhead, we chose to keep the computation of the contact force and the computation of the total force separated.

This concludes the list of functionalities that the geometric module must provide for the rest of the simulation to be performed. We describe now two potential implementations and discuss their forces and weaknesses.

4.2.2 Triangulation based on Facet-Edges

Inspired by the previous work of Xue (1995) and Müller (1996a), we have used *doubly connected facet-edge lists* (DCFL) introduced by Dobkin and Laszlo (1989). They are a natural extension of the *doubly connected edge lists* (DCEL) of Guibas and Stolfi (1985) commonly used in 2D. Though not trivial to implement, they are very powerful since they give immediate access to all the connectivity information we require, yet allow local, constant time updates.

4.2.2.1 The basic elements

The data structure comprises three kinds of basic elements: the vertex, the edge and the facet-edge. Since we deal with weighted generating sites, the vertex is defined as three coordinates and a weight:

```
class Vertex
  double      x, y, z
  double      w
end
```

The edge connects two vertices. It is oriented, thus has a starting and an ending vertex:

```
class Edge
  Vertex      start
  Vertex      end
end
```

The facet-edge is the core element of the DCFL, as it holds the complete topological structure of a 3D triangulation. A facet-edge, as the name implies, is the combination of a facet and an edge of the triangulation. It follows from this choice that every facet in the triangulation will be represented by three facet-edges and every edge will be represented by as many facet-edges as facets (or tetrahedra) there are incident to it.

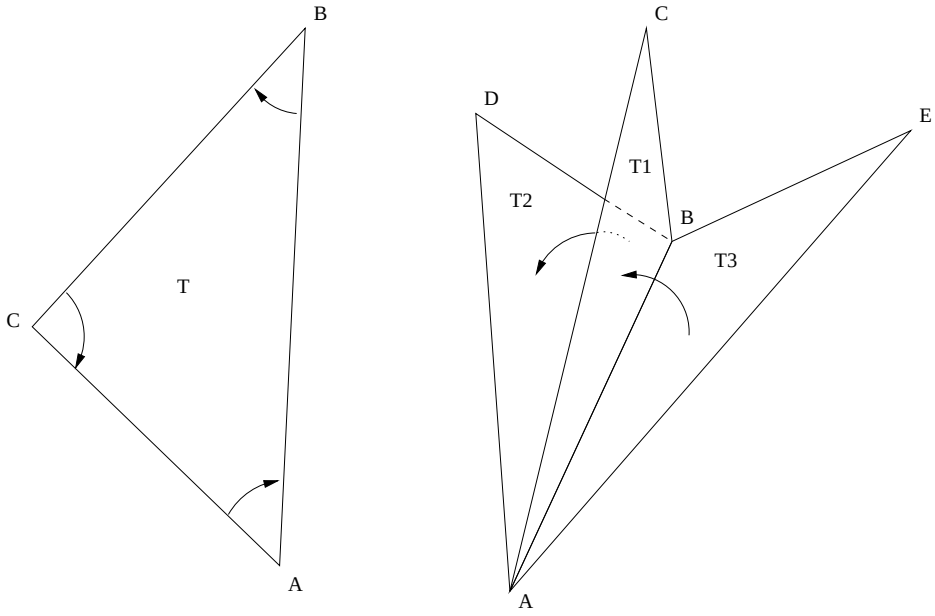


Figure 4.3: *Left:* Connections attached to a facet-edge inside a facet: T_{AB}.`Enext` gives T_{BC} while T_{AB}.`Eprev` gives T_{CA}. *Right:* Connections attached to a facet-edge around an edge: $T1_{AB}$.`Fnext` gives $T2_{AB}$ while $T1_{AB}$.`Fprev` gives $T3_{AB}$.

The facet-edges are connected together in two different ways. Within a facet, each facet-edge gives direct access to the two others through `Enext` and `Eprev` operations. Around an edge, `Fnext` and `Fprev` operations provide traversal through of all of them. Figure 4.3 illustrates those double links.

Facet-Edges are oriented, in that they have a starting Vertex and an ending Vertex. A boolean flag indicates if this orientation corresponds to that of the underlying edge. Thus, a facet-edge is implemented by the following structure:

```
class FacetEdge
  Edge       edge
  Boolean    dir
  FacetEdge  Enext, Eprev
  FacetEdge  Fnext, Fprev
end
```

in which `Enext`, `Fnext`, `Fnext` and `Fprev` are really pointers to other facet-edges, and edge is also a pointer to the suitable edge.

Note that the `Eprev` and `Fprev` pointers are somehow redundant, as it is possible to visit all neighbors by continuously moving forward. However, they are present for efficiency.

Aside from direct access to the components of the structures described so far, a number of access functions are implemented as an interface to higher-level objects. In particular, a facet-edge T provides direct access to the five vertices of the two tetrahedra sharing its associated facet, see figure 4.4.

- T.`V1` is the starting vertex of the underlying edge if it is oriented like T, the ending vertex if the orientations differ.
- T.`V2` is the ending vertex of the underlying edge if it is oriented like T, the starting vertex if the orientations differ.
- T.`V3` = T.`Enext.V2` = T.`Eprev.V1` is the third vertex of the facet.
- T.`V4` = T.`Fnext.Enext.V2` = T.`Fnext.Eprev.V1` is the fourth vertex of the tetrahedron on the *positive* side of the facet.
- T.`V5` = T.`Fprev.Enext.V2` = T.`Fprev.Eprev.V1` is the fourth vertex of the tetrahedron on the *negative* side of the facet.

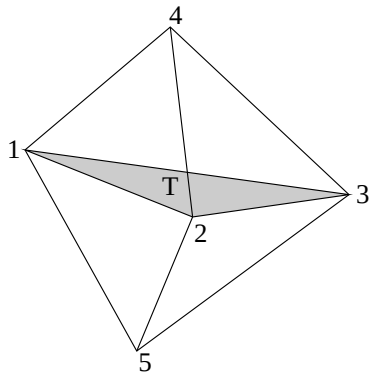


Figure 4.4: Accessing vertices from a facet-edge.

One can deduce that a tetrahedron is uniquely identified by a facet-edge and an orientation.

As mentioned before, we start with four auxiliary vertices defining a tetrahedron large enough to contain the whole simulation area, and then insert vertices inside it. For this initial triangulation, four vertices, six edges and twelve facet-edges are required.

4.2.2.2 The insertion of a new vertex

The insertion of a new vertex follows the algorithm presented in §4.2.1. The directed traversal of the triangulation is easy to perform by a succession of calls to the `Enext`, `Eprev`, `Fnext` and `Fprev` primitives, first to identify a separating facet of the current tetrahedron, then to move to the next one. Such a path was shown in figure 4.2. The point is then inserted as shown in figure 4.5.

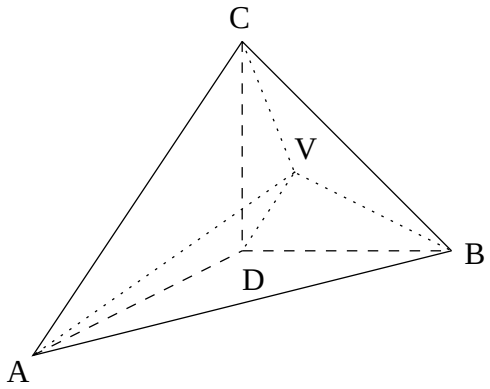


Figure 4.5: Inserting one new vertex. Four new edges {A,V}, {B,V}, {C,V} and {D,V} are created. Six new facets thus eighteen new facet-edges are also created. They are connected to each other within the newly created facets and around the newly created edges, and are inserted in the lists around the corresponding edges of the original tetrahedron {A,B,C,D}.

4.2.2.3 The flips

Once a portion of the triangulation has been identified as violating the Delaunay criterion — by means of the *insphere* predicate — it is updated by performing flips. As shown in figure 2.15, either flip reduces to adding or removing one edge and some facet-edges.

4.2.2.4 Inserting and removing a facet-edge

We conclude the description of this data structure by showing what really happens when inserting and removing facets since most operations reduce to a combination of them.

When a new facet appears in the triangulation, three new facet-edges are generated and connected to each other by the two doubly connected lists within that facet, thus setting the `Enext` and `Eprev` fields of the facet-edges. Then each facet-edge is inserted in the two doubly connected lists of facet-edges around their supporting edges, thus setting their `Fnext` and `Fprev` fields. The removal of a facet basically involves performing the same operation backwards: the facet-edges are removed from the lists of their supporting edges, then destroyed, thus destroying the facet. The insertion and removal of a facet-edge around an edge are simple procedures (see figure 4.6):

```
procedure InsertFacetEdge(T1, T2)
  # T1 is the new FacetEdge to be inserted before T2
  # Order of the operations is important !
  T2.Fnext.Fprev = T1          # T3.Fprev = T1, used to be T2
  T1.Fprev = T2
  T1.Fnext = T2.Fnext          # T1.Fnext = T3
  T2.Fnext = T1                # used to be T3
end
```

```
procedure RemoveFacetEdge(T1)
  T1.Fprev.Fnext = T1.Fnext    # ie T2.Fnext = T3
  T1.Fnext.Fprev = T1.Fprev    # ie T3.Fprev = T2
end
```

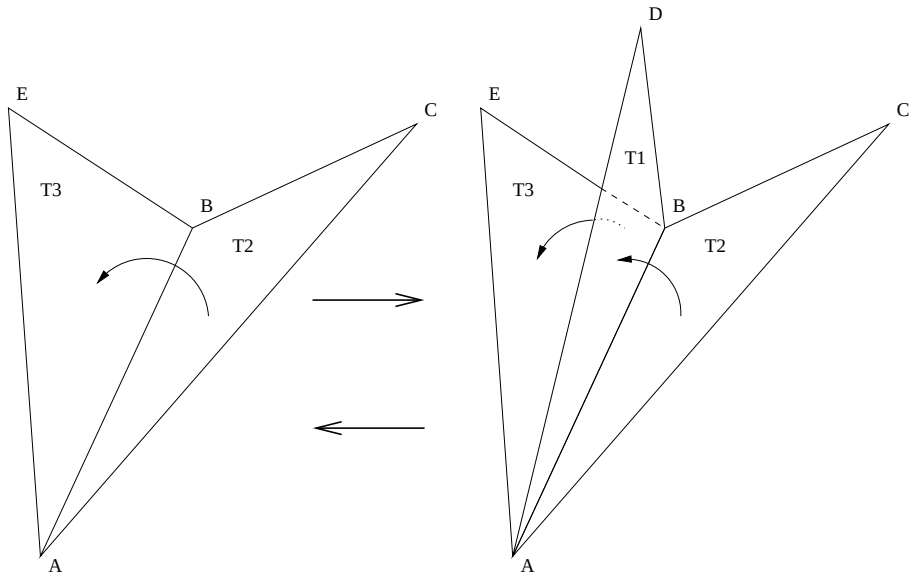


Figure 4.6: Insertion and removal of a facet-edge in the lists of facet-edges around the edge {A,B}.

4.2.3 Triangulation based on Triangles

An alternative to the DCFL was also implemented to store the triangulation. Aside from the Vertex and the Edge, the third basic element of this approach is the Triangle. Each triangle is uniquely defined by its three vertices, but also keeps a pointer to the 4^{th} vertex of both tetrahedra it belongs to (see again figure 4.4). The triangles are stored in a dictionary that provides O($\log n$) access to the set of 5 vertices indexed by the 3 base vertices. Successive retrieval thus provides complete connectivity information in the triangulation.

This approach was designed when persistent access to the triangles was necessary for drawing purposes. The support offered by the advanced data structures of the standard template library (STL) of C++ (Stroustrup, 1997) made this implementation fairly straightforward. It was however not used much because the retrieval of connectivity information from the dictionary, however efficient, is still more time consuming than with the DCFL.

Operations on the triangulation are relatively easy to implement with this representation. The insertion and deletion of facets maps to the creation and deletion of a triangle, and the connectivity is maintained by suitably updating the 4^{th} and 5^{th} vertex referenced by each triangle.

Comparing the DCFL and the dictionary of triangles yields expected conclusions: the former is more mathematical, designed for the problem at hand, fairly complex to understand and

implement, and extremely efficient. The latter is more computational, based on general-purpose data structures and functionalities, relatively easy to understand and implement, and provides poor performance. Efficiency has its cost.

4.3 Numerical stability in Computational Geometry

The major problem with the geometric predicates described in section 2.3 is that their numerical evaluation with standard limited precision floating-point arithmetic commonly provided by today's computers is very error prone, and that false evaluation of the predicates usually opens the door to all kinds of problems in the triangulation (inconsistencies in the data structures) and in the simulation (contacts not detected).

The correct evaluation of the sign of any of the determinants (2.5) to (2.12) is especially crucial when the points are "close" to being in degenerate position. In those cases, the absolute value of the determinant will be very small, since we are in fact computing a volume. And it is precisely when that value is close to zero that rounding errors are likely to happen, as the determinant is computed by adding and subtracting products of coordinates, that is, adding and subtracting large numbers whose sum or difference may have a very small modulus. Many significant digits of the operands are lost in such computations.

Efficient exact evaluation methods for these determinants have been proposed for restricted cases with integer or rational input, see for example Bronnimann and Yvinec (1997). For generic input, libraries for extended precision arithmetic exist, but they usually slow down computation by one or more orders of magnitude and are not suited for our purposes. The solution we used comes from Shewchuk (1997) and is based on two advanced techniques.

The first is a set of efficient algorithms for performing elementary operations (+, -, *) with arbitrary precision by relying on features of the IEEE 754 floating-point standard. It is inspired by the previous work of Priest (1991).

The second uses the fact that we only need the sign of those determinants, not their exact values. Therefore, it is often possible to guarantee an exact result for the predicate without an exact computation. Fortune and Van Wyk (1993) suggested using standard arithmetic to compute the determinant Δ and a certificate of validity τ. If the latter cannot assert that Δ has the correct sign, then an exact computation is performed by means of extended precision arithmetic. Shewchuk improved this scheme by computing a sequence of successively more accurate approximations of the determinant.

The idea of computing such arithmetic filters for the predicates is also used in CGAL. Our implementation is based on Shewchuk's code[2], slightly extended to support the weighted version of the Orient3D predicate. Practical usage has shown that the first approximation can be validated in more than 99% of the cases, thus achieving computational efficiency, and that the configurations where this approximation is wrong are very rare, but they do indeed happen and are fully detected, thus achieving geometric exactness.

[2] Available from `http://www.cs.cmu.edu/~quake/robust.html`.

We describe summarily the exact floating-point computations and the adaptive sign evaluation proposed by Shewchuk. The rest of this section is largely inspired by his papers. We have used the same names and symbols so that the interested reader can pursue the exploration by reading either the short report (Shewchuk, 1996) or the complete paper with detailed proofs (Shewchuk, 1997).

4.3.1 Exact floating-point computations

Floating-point numbers are represented in modern processors as a sign bit, a significand of p bits, and a base-2 exponent: $x = \pm bbbb\cdots(p\text{bits})\cdots bbbb \times 2^{\text{exponent}}$. In double precision, $p = 53$ which translates to roughly 15 significant decimal digits (see Goldberg, 1991, for a complete survey of floating-point formats). Rounding errors occur when p bits are not enough to express the result of a computation. For example, if $p = 4$, 11010 + 101.1 gives 11110 instead of 11111.1 and two digits are lost. The solution consists in having more than p bits to represent the result. Instead of allocating very long vectors of bits ($p \gg 1000$), the idea is to use *expansions*: $x = x_n + \cdots + x_2 + x_1$ where each x_i is a normal floating-point number. Two conditions are imposed: the x_i are ordered by magnitude (x_n largest, x_1 smallest) and non-overlapping, in the sense that the least significant bit of x_{i+1} is more significant than the most significant bit of x_i. Note that several expansions may represent the same value and that x_n provides immediate access to an approximation of x. In particular, if $x_n \neq 0$, x and x_n have the same sign.

Basic arithmetic operations can be performed with expansions, but before we describe them, let us see how addition, subtraction and multiplication of two standard floating-point numbers can return an exact value represented by an expansion. We use the symbols $\oplus$, $\ominus$ and $\otimes$ for the p-bit operations performed by the processor. The following algorithms assume that these operations are performed with *exact rounding*: if the result of a computation can be expressed with p bits, then the result is exact; if it cannot, the result is rounded to the nearest p-bit floating point value.

If a and b are two p-bit floating-point numbers such that $|a| \geq |b|$, `FAST-TWO-SUM` produces an expansion $x+y$ such that $a+b = x+y$, where x is an approximation of $a+b$ and y accounts for the rounding error.

```
procedure FAST-TWO-SUM(a, b)
  x = a ⊕ b
  bv = x ⊖ a
  y = b ⊖ bv
return (x,y)
```

`TWO-SUM` lifts the hypothesis on $|a| \geq |b|$:

```
procedure TWO-SUM(a, b)
  x = a ⊕ b
  bv = x ⊖ a
  av = x ⊖ bv
  br = b ⊖ bv
  ar = a ⊖ av
  y = ar ⊕ br
return (x,y)
```

TWO-DIFF and FAST-TWO-DIFF are implemented similarly. To account for the propagation of the rounding error y one must now be able to add two expansions. Let e and f be two expansions of length m and n. Simply merging the two gives the correct result, but breaks the non-ovelapping condition. LINEAR-EXPANSION-SUM starts with the merged expansions and produces an expansion h for the result in linear time:

```
procedure LINEAR-EXPANSION-SUM(e, f)
  Merge e and f in a sequence of non-decreasing magnitude g
  (Q_2, q_2) = FAST-TWO-SUM(g_2, g_1)
  for i = 3 to m+n
    (R_i, h_{i-2}) = FAST-TWO-SUM(g_i, g_{i-1})
    (Q_i, q_i) = TWO-SUM(Q_{i-1}, R_i)
  h_{m+n-1} = q_{m+n}
  h_{m+n} = Q_{m+n}
return h
```

Multiplication is based on the fact that a p-bit floating-point processor will multiply without errors two $\lfloor \frac{p}{2} \rfloor$-bit values. The exact product of two p-bit floating-point numbers a and b is performed by splitting each term, performing four exact multiplications, and reconstructing an expansion $x+y=ab$ from the four results. Full details on the splitting, the TWO-PRODUCT procedure and how it can be used to multiply two expansions are given in the original papers of Shewchuk.

4.3.2 Adaptive sign computation for determinants

Looking at the code of FAST-TWO-SUM or TWO-SUM above, one sees that the first line computes the approximation x and the remaining lines compute the rounding error y. If the precision of x is sufficient, one could avoid computing the unnecessary y, or at least delay it until more precision is really required.

Consider the two-dimensional orientation predicate given by

$$\mathrm{Or2D}(\mathrm{A},\mathrm{B},\mathrm{C}) \quad = \quad sign \begin{vmatrix} \mathrm{A}_x - \mathrm{C}_x & \mathrm{A}_y - \mathrm{C}_y \\ \mathrm{B}_x - \mathrm{C}_x & \mathrm{B}_y - \mathrm{C}_y \end{vmatrix} \tag{4.1}$$

The coordinates of A, B and C are supposed to be p-bit floating-point numbers. After a first round of computation, one has the following expansions of length two:

$$\begin{aligned} q_2 + q_1 &= \mathrm{A}_x - \mathrm{C}_x \\ r_2 + r_1 &= \mathrm{A}_y - \mathrm{C}_y \\ s_2 + s_1 &= \mathrm{B}_x - \mathrm{C}_x \\ t_2 + t_1 &= \mathrm{B}_y - \mathrm{C}_y \end{aligned}$$

Recall that $\varepsilon|q_2| \geq |q_1|$ (and likewise for the three others) with $\varepsilon = 2^{-p}$ as the expansions have non-overlapping components.

The next round produces two expansions of length four:

$$\begin{aligned} \sum_{i=1}^{4} x_i &= (\mathrm{A}_x - \mathrm{C}_x)(\mathrm{B}_y - \mathrm{C}_y) \\ &= (q_2 + q_1)(t_2 + t_1) \\ &= q_2t_2 + q_2t_1 + q_1t_2 + q_1t_1 \\ \sum_{i=1}^{4} y_i &= (\mathrm{B}_x - \mathrm{C}_x)(\mathrm{A}_y - \mathrm{C}_y) \\ &= (s_2 + s_1)(r_2 + r_1) \\ &= s_2r_2 + s_2r_1 + s_1r_2 + s_1r_1 \end{aligned}$$

Note that the previous equations do not mean that $q_2t_2 + q_2t_1 + q_1t_2 + q_1t_1$ is a valid expansion for x as these terms may overlap. However, q_2t_2 can be chosen to be x_4. Before fully constructing the two expansions x and y, we see that

$$\begin{aligned} \varepsilon\,|q_2t_2| &\geq |q_2t_1| \\ \varepsilon\,|q_2t_2| &\geq |q_1t_2| \\ \varepsilon\,|q_2t_1| &\geq |q_1t_1| \\ \varepsilon\,|q_1t_2| &\geq |q_1t_1| \\ \varepsilon^2\,|q_2t_2| &\geq |q_1t_1| \end{aligned}$$

$q_2t_2 - s_2r_2$ is a first approximation of the value of the determinant (4.1). If its modulus is large enough, its sign will not change when adding the rest of the expansions x and y. In fact, if

$$|q_2t_2 - s_2r_2| > A(|q_2t_2| \oplus |s_2r_2|) \tag{4.2}$$

then $q_2t_2 - s_2r_2$ gives the correct sign for the determinant (4.1).

The factor A in equation (4.2) is a bound for the forward propagation of rounding errors due to the computation of $q_2t_2 - s_2r_2$ instead of the exact value of the determinant, which is given if test (4.2) fails by a last expansion of length eight:

$$\begin{aligned} \sum_{i=1}^{8} \Delta_i &= (\mathrm{A}_x - \mathrm{C}_x)(\mathrm{B}_y - \mathrm{C}_y) - (\mathrm{B}_x - \mathrm{C}_x)(\mathrm{A}_y - \mathrm{C}_y) \\ &= \sum_{i=1}^{4} x_i - \sum_{i=1}^{4} y_i \end{aligned}$$

4.4 The simulation loop

As can be seen in figure 4.1 on page 46, the actual implementation of the DEM algorithm presented in section 3.2 takes place in a specialized class that accesses all the relevant information through the container. The core of this class simply performs the requested number of iterations, checks for collisions along the edges of $\mathcal{DT}$ and with the walls, and then updates the positions of the grains. This loop is also responsible to call at given intervals the triangulation maintenance routine, the statistics computation, the instantaneous exportation of snapshots of the simulation, and a few other minor functionalities.

The sequential simulation loop is thus a straightforward implementation of the algorithm. The benefit of the layered design discussed in section 4.1 is that this sequential loop can be replaced by a parallel loop without any change in the rest of the code. Using this feature, we wrote two other loops, one specialized for SGI parallel machines using their proprietary multiprocessing compilers and libraries, the other implemented with standard POSIX threads compatible with virtually any parallel machine running some UNIX-like operating system. In particular, low cost dual- and quad-Pentium machines running Linux have proven to be very cost efficient machines to run our simulations. The two parallel loops differ only in their implementation syntax. We shall now describe the steps required to go in parallel.

4.5 The parallel simulation loop

4.5.1 The parallel algorithm

The DEM algorithm of section 3.2 is well suited to a medium grain[3] parallelism. Each iteration is divided in two main phases:

- In step 3, compute the forces along each edge of the triangulation as a function of the position (and velocity, spin,...) of the grains.
- In step 4, compute the positions (and velocity, spin,...) of the grains as a function of the forces in the incident edges and external forces.

These two phases account for the major part of the computational effort. As mentioned before, they cannot be interleaved, due to data dependencies. However, each phase exhibits a very high degree of parallelism since each force, resp. the position of each grain, may be computed independently. It is fairly easy to divide the simulated area into layers and assign the grains and edges in a layer to a given processor, as shown in figure 4.7. That processor is then responsible for updating all values in that layer, eventually reading values from a neighbor layer.

[3]The term grain here has nothing to do with granular materials. It is a measure of the size of the blocks of code that are distributed over the processors. In fine grain parallel codes, very small portions are executed with a very tight synchronization. In coarse grain parallel codes, very large procedures are executed by almost independent processors. Medium grain codes are somewhere in-between.

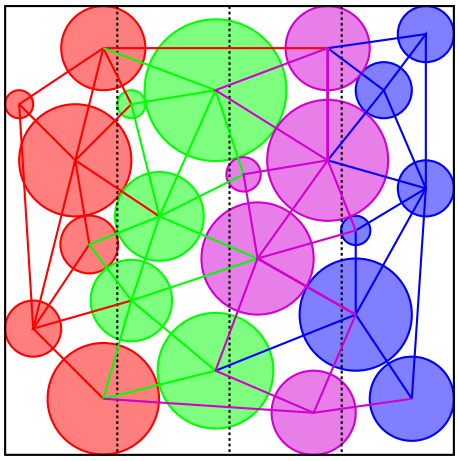


Figure 4.7: The simulated area is divided in layers that are assigned to processors.

Since the workload associated with each layer is directly proportional to the number of elements, the layers will be chosen such that all of them contain an approximately equal number of vertices and edges. If grains are added during the simulation, it is necessary to reform the layers in order to rebalance workload, as shown in figure 4.8. Even without adding new grains, it is better to reform the layers now and then in order to minimize the number of accesses to elements of neighboring layers. The assignment of grains and edges to the processors is easily achieved as part of the triangulation update in step 5.

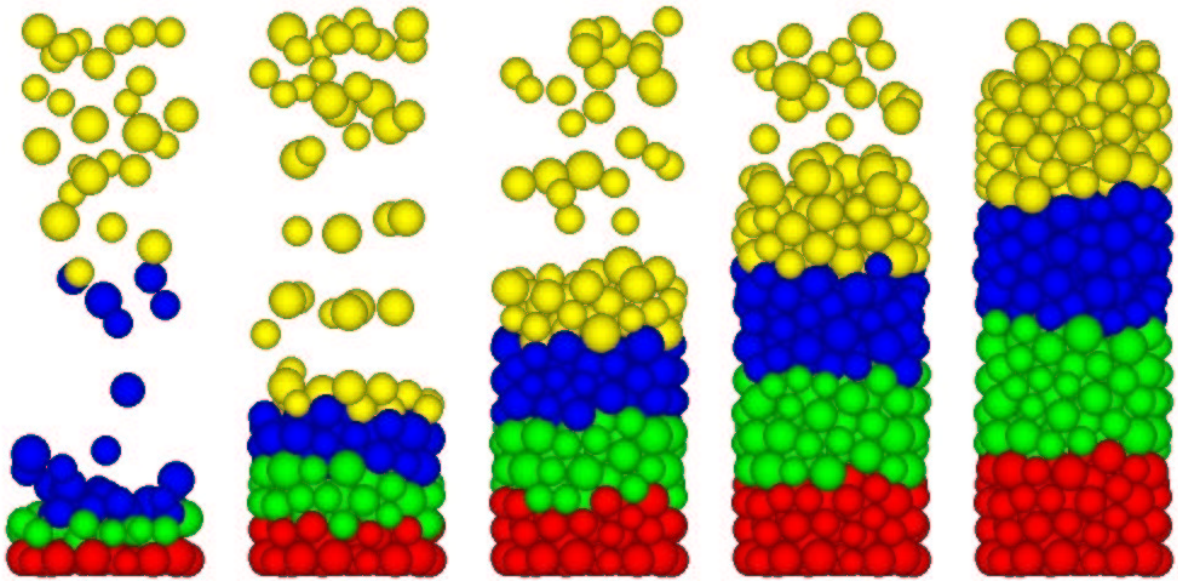


Figure 4.8: Five consecutive snapshots of grains being poured in a cylinder. The color of a grain indicates which of the four processors is responsible for computing its trajectory. The dynamic subdivision into layers implies in this case that the horizontal boundaries vary with time.

Most of step 5 deals with checking the validity of the triangulation through the computation of geometric predicates. That can again be performed in parallel for each layer. The short phase of topological changes and layers redistribution that takes place if the triangulation is not valid

anymore is performed in a sequential block, as is the initial construction of the triangulation (step 1), the computation of the various statistics and the exportation of the simulation results.

Figure 4.9 gives the pseudo-code for the parallel version of the algorithm. Steps performed by all processors in parallel and steps performed by only one processor are distinguished, and the necessary synchronization points between the various phases (barriers) are shown.

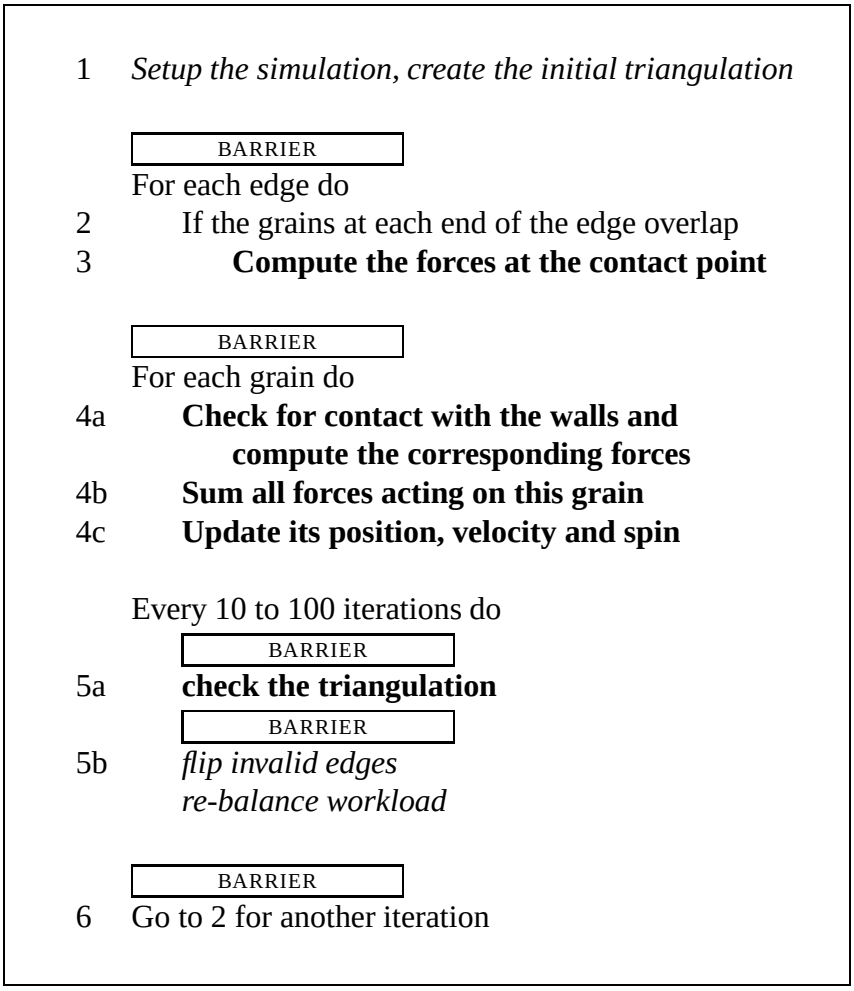


Figure 4.9: The pseudo-code of the parallel version of the algorithm. Steps in **bold** are computed in parallel by each processor, while steps in *italics* are done sequentially by only one processor. The synchronization points (BARRIER) also appear.

4.5.2 The parallel machines

The major drawback of this parallel algorithm is that it relies on transparent remote memory accesses. Transparent means here that if in order to update an element assigned to it, processor A needs data from an element assigned to processor B, A must be able to read that data with no explicit intervention of B. Such remote memory accesses can be provided intrinsically by the hardware, or can be simulated by specialized hardware and software.

Truly shared memory hardware, such as SMP PCs or workstations with few processors, offers uniform memory access to every processor. In other words, accessing data from elements in neighboring layers does not cost any additional overhead[4]. Unfortunately, such hardware does not scale well beyond 8 processors.

Shared memory can also be provided by intrinsically distributed hardware in so-called single memory image, Non Uniform Memory Access (NUMA-SMP) machines such as the SGI Origin servers with 8 or more processors. However, in this case remote memory accesses may be several orders of magnitude slower than local accesses, thus maximizing data locality is crucial. Indeed, previous experience of the author gained in a different project (about the parallel computation of graph diameters, see Ferrez et al., 1998a,b) has shown that the performance drastically decreases if one blindly trusts the shared memory scheme advertised by these machines instead of accounting explicitly in the algorithm for the distributed hardware.

The Cray T3D, one of the precursor of the massively parallel programming paradigm, did not offer shared memory but customized hardware and a specific library, `shmem`, provided efficient and transparent remote memory access, exactly what we needed to parallelize the two dimensional simulation code of Müller, see Ferrez et al. (1996). The reduced complexity of the 2D triangulation also made things much easier. In particular, with a fixed convex hull and as long as no new grain is added in the simulation, the number of edges of the triangulation remains constant. In 3D, every flip involves creating and deleting edges and facets, thus rendering the memory management more tedious.

The current trend in High Performance Computing, however, is clusters of machines that do not provide any remote memory access[5]. In this case, the data structures and algorithms must be adapted to account for explicit exchanges of information between two processors commonly performed in the framework of the Message Passing Interface (MPI). Indeed, a processor that requires remote data will only receive it if the remote processor knows about this requirement and explicitly transmits the information. Minimizing remote accesses is even more crucial in this case, not just because the communication bandwidth is generally smaller, but because latencies are up to several orders of magnitude higher and the communication actually interrupts both processors.

The DCFL data structure, already fairly complex in a sequential algorithm, becomes tedious to maintain in a message-passing context. An efficient, distributed memory implementation of the dynamic triangulation will require careful attention and eventually an alternative data structure.

Another possible approach would be to differentiate the role of the processors: one of them, the master, would take care of the whole triangulation and do nothing else, while the others, the slaves, would compute forces and trajectories in the layers. This model implies large volume of communications between the master and the slaves — all the grains coordinates must be exchanged every time the triangulation must be updated — thus creating a significant bottleneck that might also restrict scalability to only a few processors.

Aside from reducing computation time, parallel machines are sometimes required to satisfy

[4] We assume here that contentions due to concurrent memory accesses to a read-only variable have no impact. This is true for the validity of the method, but may indeed influence the memory throughput and therefore the overall performance.

[5] Software emulations exist (see for example Carreira et al., 1995) but at the price of very poor performances.

huge memory needs. This is not a problem with the current models since a standard workstation with 512Mb of memory will host a simulation of 100'000 spherical grains. However, more complex geometries or contact models that keep a history of the collision might change this situation.

4.5.3 Performance of the parallel code

The scalability of the shared memory parallel code has been tested on an SMP PC with 4 Pentium III Xeon processors. The simulation comprised approximately 12'000 grains of three different sizes packed in a cylinder. 1000 iterations were performed with the triangulation being checked and updated every 10 iterations. Running times and speedup compared to the sequential code running on the same machine are given in table 4.1 and figure 4.10.

Step of the algorithm		**1 CPU**	**2 CPUs**	**3 CPUs**	**4 CPUs**
Setup	1	272	267	273	267
Grain-grain forces	3	450	235	181	140
Grain-wall forces	4a	106	55	48	38
Trajectories	4b&c	475	235	182	133
Triang. verification	5a	178	92	69	52
Triang. maintenance	5b	161	163	181	167
Other (export...)		128	83	75	63
Total time		1770	1130	1009	860
Speedup		—	1.56	1.75	2.06
Total time (w/o setup)		1498	863	736	593
Speedup (w/o setup)		—	1.73	2.04	2.53

Table 4.1: Timing of the shared memory parallel code. Times are given in seconds of wall clock time, averaged over three runs that gave results within at most 2%.

These values call for some comments. First of all the setup time may seem very long, taking approximately 15% of the total time on one CPU. This incompressible sequential portion would have less influence if more iterations had been performed. Total times and speedup without this part are provided to give a better idea of what happens on long simulations.

The sequential triangulation maintenance further limits the scalability. If longer intervals are chosen, such as every 20 or 50 iterations, the impact of this step on the overall computation time, especially on several processors, is reduced.

The computation-intensive steps (3, 4a,b,c and 5a) scale almost linearly on 2 CPUs and resonably well on 3 and 4 CPUs, where memory contention effects appear. The triangulation maintenance is again sequential, and no explanation could be found for the degraded performance of this step on 3 CPUs in all three runs.

To conclude, despite the fact that the lack of a parallel code adapted to message-passing prevents us from using machines with more processors, we are satisfied with our approach that makes the most of hardware nowadays commonly found on the desktop of any computational

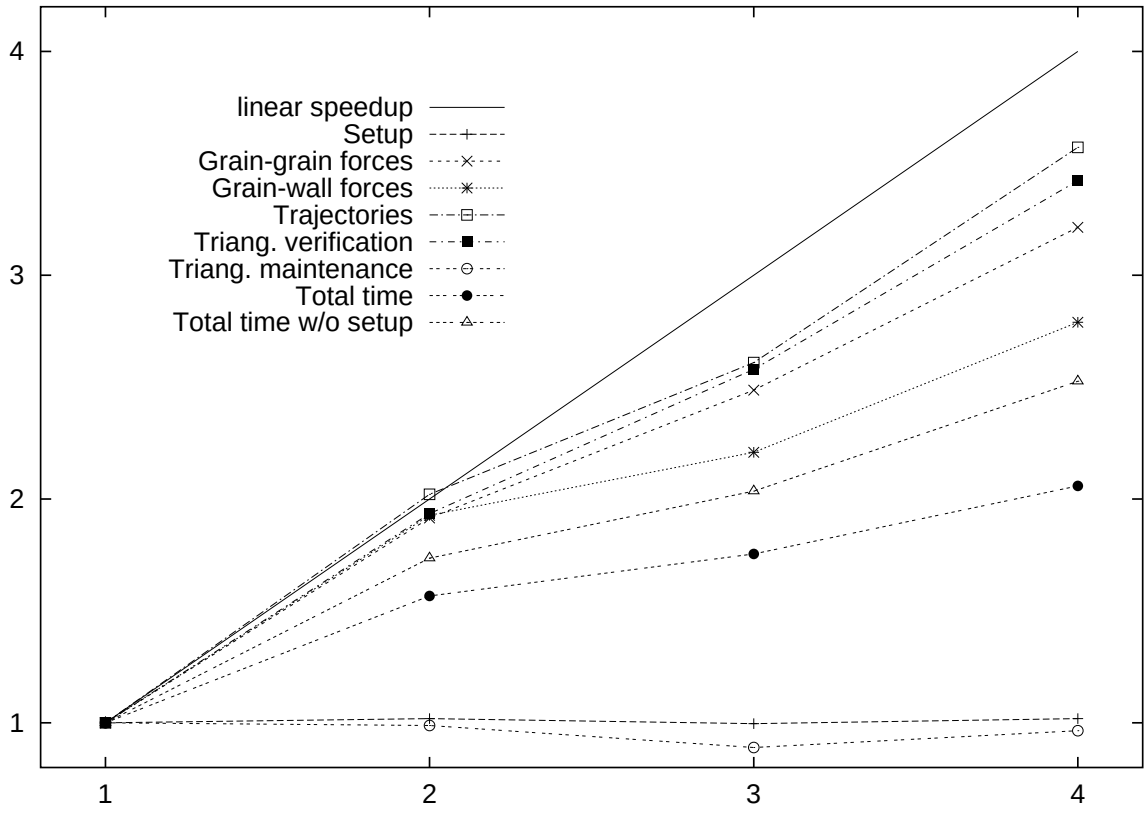


Figure 4.10: Speedup of the shared memory parallel code.

scientist. Furthermore, as the parallel versions are obtained by selecting the relevant component of the modular code, all versions remain synchronized and benefit from all improvements. The traditional approach of forking the development to produce a specialized parallel version, followed by the back-porting of new features, is avoided.

4.6 Measures

Characteristics about the simulations may be computed every few iterations as functions of position, velocity, size, etc. of the particles. For example, the average density of a packing of homogenous grains in a vertical cylinder at any time is given by:

$$\text{density} = \frac{\sum_i \frac{4}{3}\pi r_i^3}{B\, H_M} \tag{4.3}$$

where B is the surface of the base of the cylinder and $H_M = \max\{H_i\}$ is the height of highest grain. If the number of grains remains constant, only H_M need to be tracked (during step 4 of the algorithm), the rest may be precomputed for efficiency. Other global measures include kinetic, potential and total energy, maximal and average velocity or spin of a grain, etc.

Aside from global measures, local or internal measures can be performed. When suitably chosen, these values give a detailed and precise description of the actual behavior of the assembly.

Furthermore, they are usually not observable in a lab experiment. Examples include local densities per horizontal layers used to track wave effects induced by vibrations, average flow through a given area,...

Horizontally layered densities are fairly straightforward to compute with equations

$$\text{VolBelow}(G_i,h) = \begin{cases} 0 & \text{if } h \leq y_i - r_i \\ \frac{\pi}{3}(h-y_i+r_i)^2(2r_i-h+y_i)) & \text{if } y_i - r_i < h \leq y_i \\ \frac{2\pi}{3}r_i^3 + \frac{\pi}{3}(y_i-r_i-h)^2(4r_i-y_i+h) & \text{if } y_i < h \leq y_i + r_i \\ \frac{4\pi}{3}r_i^3 & \text{if } y_i + r_i < h \end{cases} \tag{4.4}$$

for the volume of the portion of grain G_i of radius r_i and vertical coordinate y_i under the plane of altitude h and

$$\text{VolInSlice}(k,\Delta h) = \sum_i \text{VolBelow}(G_i,k\Delta h) - \sum_i \text{VolBelow}(G_i,(k-1)\Delta h) \tag{4.5}$$

for the volume of the portions of all grains in the k^{th} horizontal slice of thickness Δh.

With many experiments taking place in vertical cylinders, the computation of vertical concentric layered densities is of uttermost interest. Unfortunately, such a computation requires the volume of the intersection of a sphere with a cylinder, a quantity more easily defined than computed:

$$\text{VolIntSphCyl}(r,R,d) = 2\int\int_{\substack{(x-d)^2+y^2\leq R^2 \\ x^2+y^2\leq r^2}} \sqrt{r^2-x^2-y^2}\,dx\,dy \tag{4.6}$$

where r is the radius of the sphere, R the radius of the cylinder, and d the distance between the center of the sphere and the axis of the cylinder. If the intersection is not empty, that is if $d < r+R$, this integral can be rewritten as

$$\text{VolIntSphCyl}(r,R,d) = 4\int_{\max(d-R,-r)}^{\min(d+R,r)} dx \int_0^{\min\left(\sqrt{R^2-(x-d)^2},\sqrt{r^2-x^2}\right)} \sqrt{r^2-x^2-y^2}\,dy \tag{4.7}$$

which is not easier to compute numerically. Fortunately, Lamarche and Leroy (1990) have proposed an alternative expression that can be computed directly as well as a sample FORTRAN implementation, from which we could derive a C++ version. An equation similar to (4.5) can then be used to compute local concentric densities.

4.7 I/O functionalities

The I/O functionalities provided by the simulation environment belong to two categories. *Checkpointing* allows dumping the current status of the simulation to a file and reloading it later. *Exporting* at suitable times provides a partial view of the simulation in a format that allows post-processing by another tool.

4.7.1 The checkpointing mechanism

Checkpointing is a common feature of many High Performance Computing environments. Its objective is to dump a running application and its data to disk in order to remove it from the system, thus freeing memory and CPU. On highly loaded systems, this is used to slice and distribute resources to several users. The checkpointing mechanism can be offered by the operating system or by the batch management system. Why implement our own checkpointing mechanism ? Our main objective was to be able to save the current state of the simulation and to retrieve it later, continuing the simulation, possibly with a different set of parameters.

The structure of the simulation code, with one module responsible for the storage of (lists of) elements made such a checkpointing fairly simple to implement: Every class has a unique identifier and is able to dump and restore itself from a stream. The container is then responsible to call every object of the simulation, telling it to dump itself. The restoring mechanism is similar, the container reads a class identifier and creates an object of the correct type, the object being initialized with data coming from the stream.

4.7.2 The exportation of data for visualization

An important way to extract information from a computer simulation is visualization. Unfortunately, while the choice of spherical grains simplifies the simulation, it restricts the interactive visualization possibilities as 3D spheres are among the most difficult objects to draw on a computer screen and even most recent high-end visualization servers do not allow comfortable online observation of large data sets. Two trends emerge in the transformation of the computational results into visual experience:

High-resolution images of instantaneous snapshots of the simulation can be rendered using various techniques such as ray-tracing. Eventually, series of such images are assembled into video clips. This approach is suited for static and non-interactive representation such as book or poster illustrations, web pages, etc. In fact, this report contains more than 200 images produced this way. The main drawback — aside from a considerable post-processing effort — is the lack of interactivity. It is not possible to view a certain situation under a different angle, to perform projections, clippings, boundary removal, etc. without repeating the whole process.

On the other hand, online interactive visualization is possible on small cases containing 100 to 1000 spheres. Aside from comfortably studying small cases during the development of the method, this allows for example to track only a few selected particles or to restrict the visualization to a very small portion such a the opening of a hopper. However, the lack of a global vision in these cases is a strong obstacle to the overall comprehension of the phenomenon.

4.8 Parameter management

Aside from a limited number of options that are given on the command line, among which the level of debugging information to be produced, all the parameters are read from an initialization

file. This file is meant to be as readable as possible: it is organized in logical sections, allows comments to be inserted wherever they make sense, and uses meaningful parameter names. A short example is given in figure 4.11.

```
[General]
Gravity        = (0.0, -9.81, 0.0)
nbPoints       = 100000
nbIter         = 200000
deltaT         = 1e-5
NbThreads      = 4

# The next section contains the names of the files
# used during the simulation.  Each filename is built
# by adding the corresponding extension to BaseName.
[Files]
BaseName       = "test-22"
OutputFile     = ".wdt"
# The %d will be replaced by the iteration number.
VRMLFile       = ".%d.wrl"
```

Figure 4.11: An minimal example of initialization file.

Reading such a file in a safe and efficient way is no trivial task. We have used the standard tools *flex* and *bison*, or more precisely their C++ enabled cousins *flex++* and *bison++*, to achieve this goal. *flex++* generates the code that implements the lexical analyser, while *bison++* generates the code that implements the actual parser. The latter has been used in such a way that it returns a unique *Params* class with a private member of the correct type for each parameter found in the initialization file, and public access methods that the program can use for actually retrieving the values assigned to the parameters.

The set of parameters evolves a lot during the development of such a simulation program. Every time a new parameter is added, changed or removed, the scanner and parser description files as well as the interface and implementation of the *Params* class had to be edited accordingly. This fastidious and error-prone process has been replaced by an automatic file generation scheme that produces those files from a sample initialization file. This process is sketched in figure 4.12.

The automatic generation of the four files is itself based on the same tools *flex++* and *bison++*. It supports four data types: integer and floating-point numbers, strings and three dimensional vectors, but can easily be extended to support other types.

A side effect of this process is that it became trivial to provide default values to all the parameters, so that the end user of the simulation code is only required to provide values for the parameters he wants to set, and does not need to understand every detail of every feature of the code in order to perform basic simulations.

This parameter management module is not specific to the rest of the code or even to DEM simulation. It is in fact shared with *Biogeme*, an optimization toolbox for econometric applications

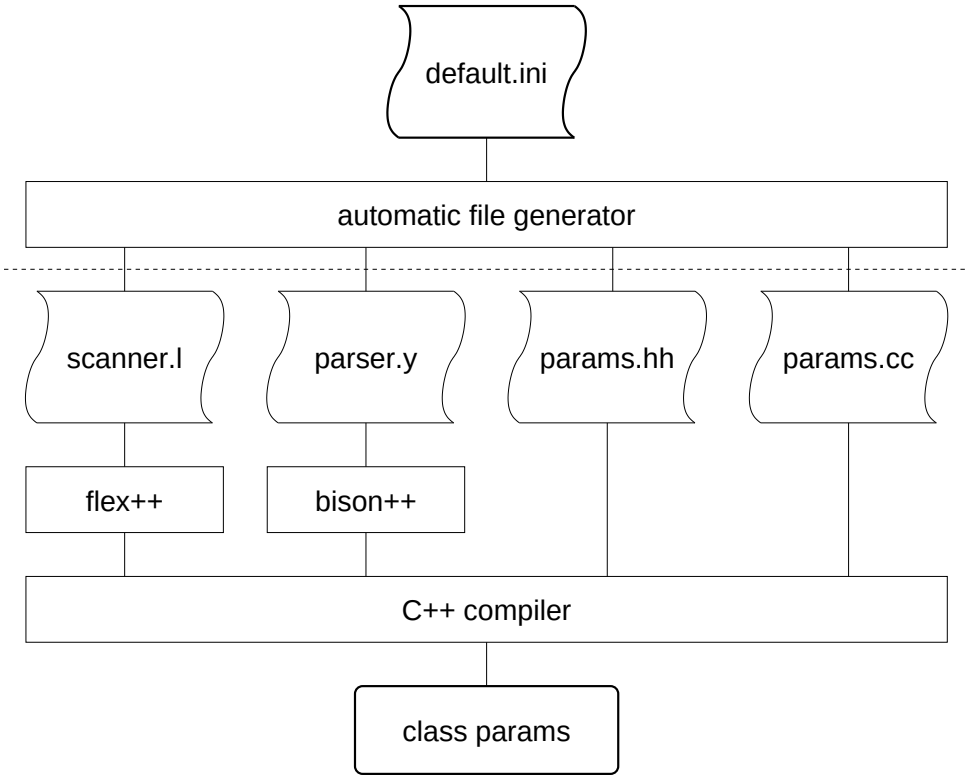


Figure 4.12: The automatic generation of the *Params* class based on a sample initialization file. Without the automatic generation scheme (above the dashed line), the programmer is required to maintain four files.

(Bierlaire, 2001), and could probably benefit other scientific codes. Two extensions are planned: a graphical user interface (GUI) for the input of the parameters and the ability to remotely control a running instance, thus allowing to modify some parameters after the computation has started.

4.9 Adding new features

We conclude the description of our implementation by presenting the framework in which new functionalities may be added to the code. As shown in figure 4.1 on page 46, this is provided for new contact models, new boundaries and new measures and exportation facilities. Writing new derived classes should be possible with only minimal knowledge of the internal structures of the whole code.

4.9.1 Integrating new contact models

A virtual base class for all the contact models provides:

- input access to the values used for computing a force at a contact point, namely the normal and tangential values of the overlap as well as their first time derivatives,

- output access for the normal and tangential components of the newly computed force,
- a virtual function `updateforce` that is called at every iteration for every contact.

The parameter management subsystem described in section 4.8 makes it trivial to introduce new parameters for the new force model in an efficient and user-friendly manner. Furthermore, it is relatively easy to extend those facilities by adding more functionalities to the new derived class.

4.9.2 Integrating new boundary shapes

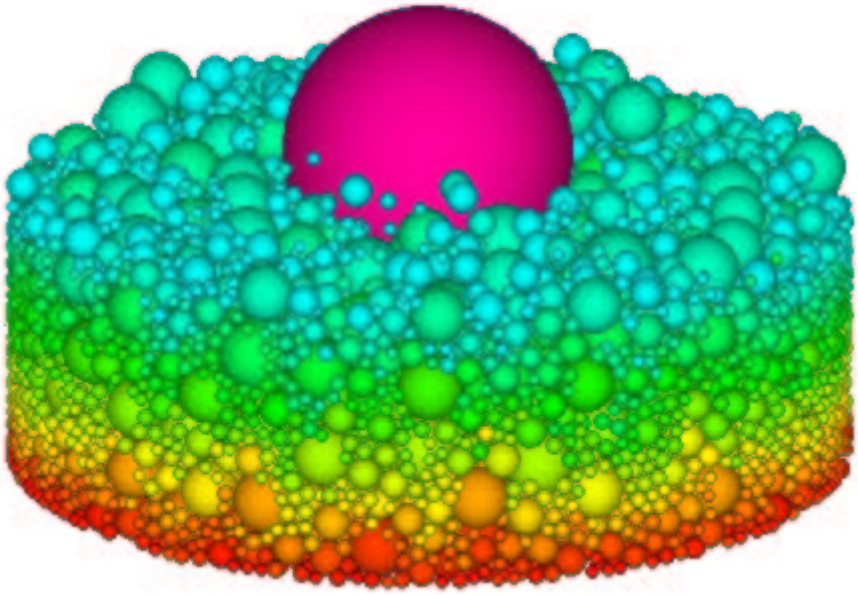
A virtual base class for all the boundary elements provides the framework for interacting with the grains. In particular, in order to implement a new shape one must provide the methods

- `distance` to compute the distance of a point in the Euclidean space E^3 to that shape,
- `normal` to provide a normal unit vector at a contact point

The existing framework will then use these methods to determine whether a given grain is in contact with a given boundary and if necessary take suitable measures to apply a force model at the contact point. The class should also provide whatever is needed for drawing the boundary. The current version of the environment contains classes for flat walls, cylindrical and conical containers, and a few other specialized items.

4.9.3 Integrating new export formats

Likewise, a virtual base class provides a framework for calling derived classes implementing exportation in various formats. An exporter may be called either every few iterations or at the end of the simulation, or both. The exporter has full access to all internal data structures of the simulation and thus can produce any sort of output. The parameter management system provides a uniform and convenient way to deal with filenames. Currently, classes exist that write VRML files for online interactive visualization, POVRay files for post-processing, producing high resolution images by ray-tracing, scene description files for the AVS visualization package. Other classes in this framework are responsible for computing statistics on the simulation, in particular the measures described in §4.6.



Part II

Applications

A designer knows he has achieved perfection not when there is nothing left to add,
but when there is nothing left to take away.
– Antoine de Saint-Exupéry

Chapter 5

Sphere packing

5.1 Introduction

An interesting application was proposed by the Energetic Material Labs of Swiss Defense Procurement Agency at Thun. The problem is to reach high densities in packings obtained by mixing three different powders. These powders are distinguished by the size of the grains and classified in fine, medium and coarse, with average diameters having ratio of 1 to 5 to 35. The general objective is to determine the optimal quantities of each type of powder — within certain limits — that result in the densest packings. We consider here only the density of such packings. Other factors like segregation, homogeneity, stress distribution, etc, are not addressed.

The mathematical model assumes spherical grains. Optimal space packings of spheres of equal size have kept physicists, chemists, geometers and mathematicians busy for several centuries. It was commonly agreed that the face-centered cubic packing[1] achieved the highest density, but no one had been able to prove this assertion, known as the Kepler conjecture and included by Hilbert in his famous list of Problems in Mathematics at the International Congress of Mathematicians in Paris in 1900:

> ”I point out the following question [...] important to number theory and perhaps sometimes useful to physics and chemistry: How can one arrange most densely in space an infinite number of equal solids of given form, e.g., spheres with given radii or regular tetrahedra with given edges (or in prescribed position), that is, how can one so fit them together that the ratio of the filled to the unfilled space may be as great as possible ?” [2]

[1] There is an infinity of equivalent optimal packings distinguished by the periodic arrangement of their layers. The face-centered cubic is one of them (with periodicity 3), the compact hexagonal is another (with periodicity 2). These two packings stand out because they play an important role in materials science (see for example Kurz et al., 1991, page 64–68), yet any rearrangement of the layers yields an density-optimal packing. For clarity’s sake, we shall improperly use the term face-centerd cubic packing to designate them all.

[2] Quoted from Proceedings of Symposia in Pure Mathematics, AMS, vol 28, 1976.

The conjecture was finally proved by Hales (1998) who used very elaborate modeling and large-scale linear programs solved by state-of-the-art methods of mathematical optimization to eliminate all other candidates.

However spectacular, this result is of little practical help for our purpose. There is already a gap between the theoretical face-centered cubic packing that yields a density of 74.018% and realistic densities of poured materials that usually yield values around 65%. The situation gets even worse for packings of grains of various sizes since there exist no theoretical results. The *a priori* calculation of the density of a packing of spheres of different radii remains a challenge. In practice, experimentation has already provided some observations of the relation between the relative proportion of each powder and the resulting density. The computer simulations are expected to reinforce these observations and provide new insights, even though we have no ambition to settle this question completely.

If we focus here on the measure of the density obtained by a given mixture, the long-term objective remains the reverse procedure: optimize the mixture for reaching a maximal density. We start by presenting the general setup in which the simulations were conducted, then present several results. The last section briefly mentions other approaches to the problem, in particular the gap between Kepler's theoretical result and actual powder packings.

5.2 The setup for the simulations

We describe in this section the global setup shared by most simulations. That setup was inspired by the experiment performed at the Energetic Material Labs of Swiss Defense Procurement Agency at Thun with the original powders. Over time, some simulations were performed with different setups. Those changes are mentioned in the description of the simulations in the following sections.

The simulations are performed in a vertical cylinder closed at the bottom and open at the top. The grains are subject to gravitation.

One way of improving the density of powder packings is to vibrate them. The idea behind this is to temporarily increase the space around the grains, thus allowing them to rearrange themselves, and by going from one locally stable configuration to another improve the global density. However, global vibrations cannot control local behavior and no promise can be made that vibrated packings will be better. Indeed, along the vibration process, the density fluctuates.

The parameters of the vibrations include frequency (usually in the 1Hz to 10Hz range), amplitude (absolute or relative to the size of the grains) and shape (sinusoidal, tapping, continuous or intermittent,...). All of them certainly have an impact on the packing process. At the Energetic Material Labs, the vibrations are generated by the eccentric rotation of a non-symmetric plate, yielding a form of tapping with a frequency of about 4Hz and an amplitude of 3mm[3]. The vibrations used in the simulations follow this pattern, see figure 5.1.

[3] A picture of the installation can be found on page 99.

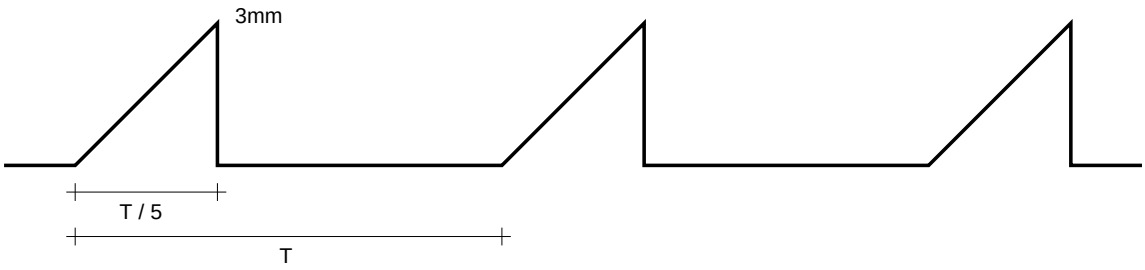


Figure 5.1: The shape of the vibration used in the simulations. T is the period.

The simulation is started by placing spheres of various radii in the cylinder. This happens in three phases, each corresponding to one category of grains: large, medium and small. The parameters for each phase i are:

- The average radius R_i of the grains, to which a small random perturbation of at most $\pm 2\%$ can be added.
- The distance D_i between two consecutive grains of phase i. Since we are using a face-centered cubic packing (see Bircher, 1998; Hales, 1998) and not just a regular grid, the actual distances between consecutive layers of grains are D_i, $\frac{\sqrt{3}}{2}D_i$ and $\frac{\sqrt{2}}{\sqrt{3}}D_i$. This allows having a higher initial density, closer to actually pouring grains, and avoids having grains pile up independently.
- A maximal filling height H_i for grains of phase i, mainly used to guarantee that the larger grains will be covered by enough smaller ones.
- A probability P_i of actually placing a grain of phase i at a given position. This avoids generating regular arrangements and provides finer control on the relative proportion of grains of each size.

Thus, some large grains are placed in the cylinder, then the remaining space is filled with medium grains, and again the remaining space is filled with small grains. A check is performed to avoid overlapping grains in the initial placement. This filling process is shown in figure 5.2. As the initial placement of the large grains has a strong influence on the packing quality, we introduced a finer control on their individual location.

The average density of the packing is computed as the proportion of the space filled by grains, and is therefore inversely proportional to the height of the granular assembly in the cylinder. Density is usually reported as a percentage, with values between 60% and 70% depending on the caliber distribution.

Due to the vibration, this measure is usually not very significant *during* the simulation, but its final value after the vibrations have stopped and the granular assembly has stabilized is the key measure for the whole process.

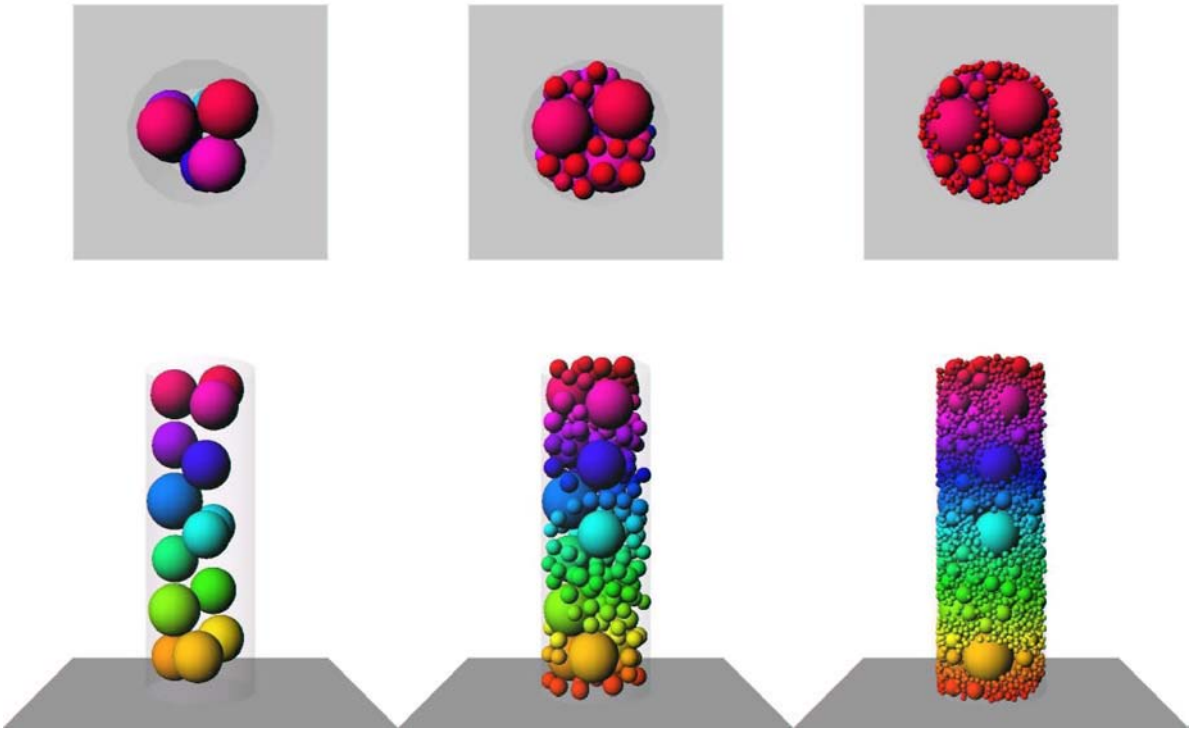


Figure 5.2: The filling process in 3 phases, top and side view. This example has 3363 grains, 14 large ($R_1 = 9$mm), 185 medium ($R_2 = 3$mm) and 3164 small ($R_3 = 1$mm). Thus, the proportion in mass is 55.6% (large), 27.2% (medium) and 17.2% (small).

Finer density measures by horizontal layers are also provided in some cases. They give a better understanding of the packing evolution during the vibrations, in particular regarding wave propagation, and are also a good indicator of strong segregation effects.

One major problem for the simulation of packings of grains with large to small ratio in the range 20 to 30 is the number of grains required to fill the space between the large grains, as shown on figure 5.3. On the other hand, when enough grains are involved, it is possible to visualize the rearrangement of medium and small grains that takes place when they leave space for the large one. Figure 5.4 gives snapshots of this phenomenon.

Various artifacts were used to circumvent this problem:

- reducing the size gap between small, medium and large grains
- truncating large grains
- focusing on local phenomena

but the current limitation on the number of grains that can be reasonably simulated remains the major obstacle preventing us from replicating exactly the original experiment.

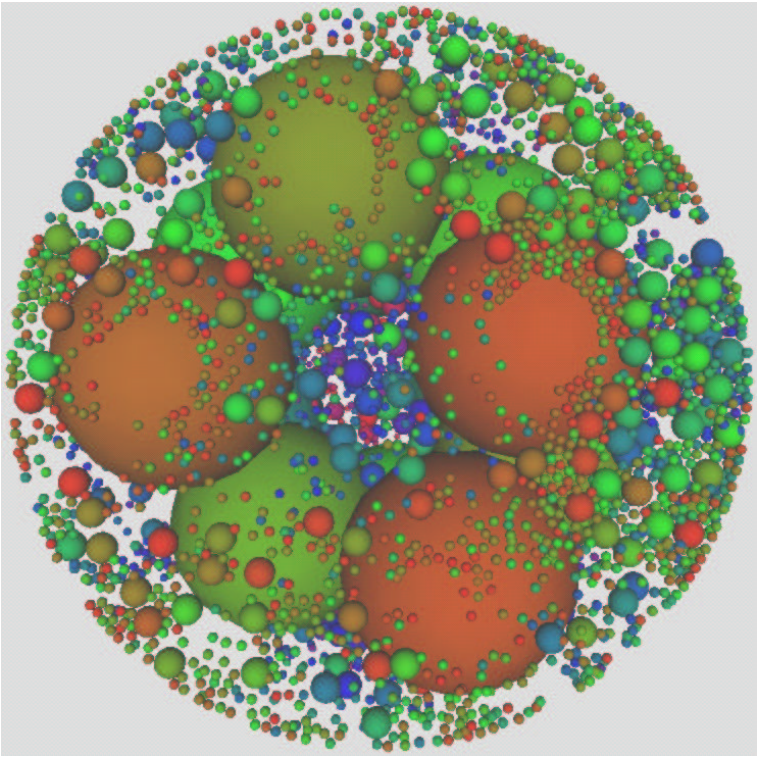


Figure 5.3: Bottom view of a simulation with a total of 4326 grains (12 large, 362 medium, 3952 small), early in the packing process. The large grains will not be covered by the others, so the measured density is too low for any practical use.

Other parameters for the simulations include the energy restitution coefficient, set to 0.3, and the friction coefficient, set to 0.1. The contact duration is 10^{-3} seconds, thus the iteration duration is $\delta t = 10^{-5}$ seconds. The simulations are performed for up to 12 seconds, so that up to 1'200'000 iterations were computed. The triangulation is updated every 100 iterations, and complete snapshots of the situation are taken at various time interval.

5.3 The first attempts

A first set of simulations involves one large grain of diameter 26mm, medium grains of diameter 3mm and small grains of diameter 1mm. Simulations with only one large grain can certainly not yield global results on the packing. However, they helped us understand how medium and small grains rearrange themselves around the large one. These cases also served as the first validation tests for the simulation method and environment. Four cases were studied, with various numbers of grains, as shown in table 5.1:

The evolution of the densities is given in figure 5.5. They can be decomposed in two phases:

1. a quadratic increase, corresponding to each grain falling down subject only to gravity,

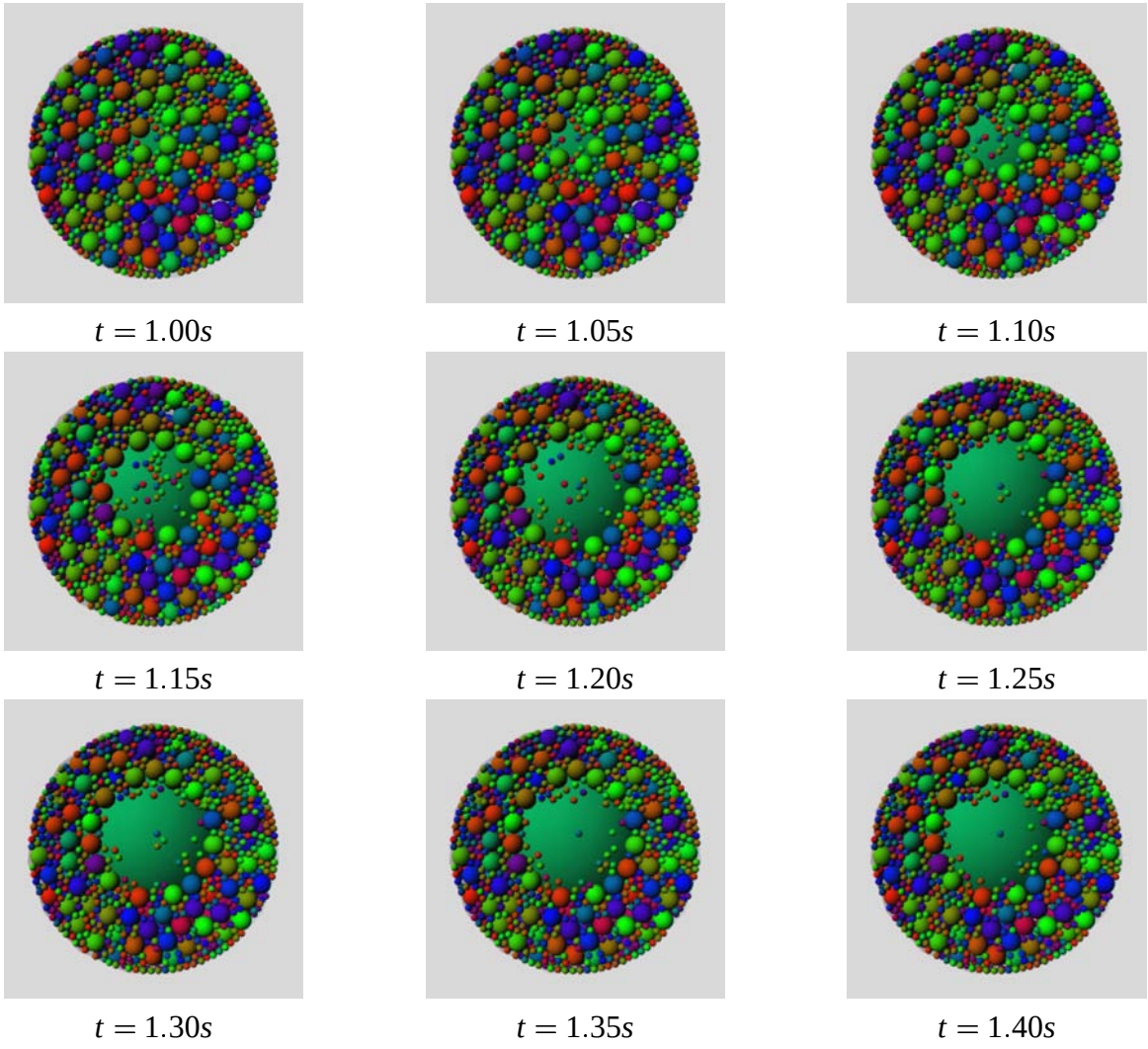


Figure 5.4: During the packing process, the medium and small grains rearrange themselves to leave space for the large one (bottom view).

		One1	**One2**	**One3**	**One4**
Large grains	(26mm)	1	1	1	1
Medium grains	(3mm)	715	378	1406	706
Small grains	(1mm)	161	498	355	1055
Total number of grains		877	877	1762	1762
Density at t=0.0s	[%]	20.1716	15.3337	30.4842	20.4489
Density at t=0.5s	[%]	63.5726	52.2700	62.7705	65.1489
Density at t=3.0s	[%]	63.5868	53.2042	64.3458	68.7132

Table 5.1: Main characteristics of the first set of simulations.

2. a slow increase, corresponding to the rearrangement of the grains. Cases One1 and One3 (more medium grains) and cases One2 and One4 (more small grains) have a similar behavior.

Case One2 is not very significant with only single-value density computation, since the large

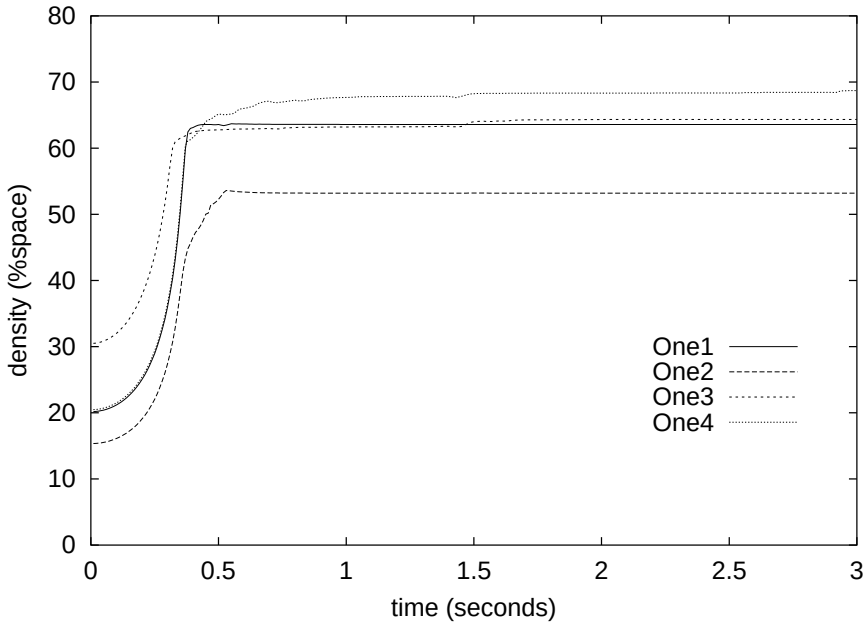
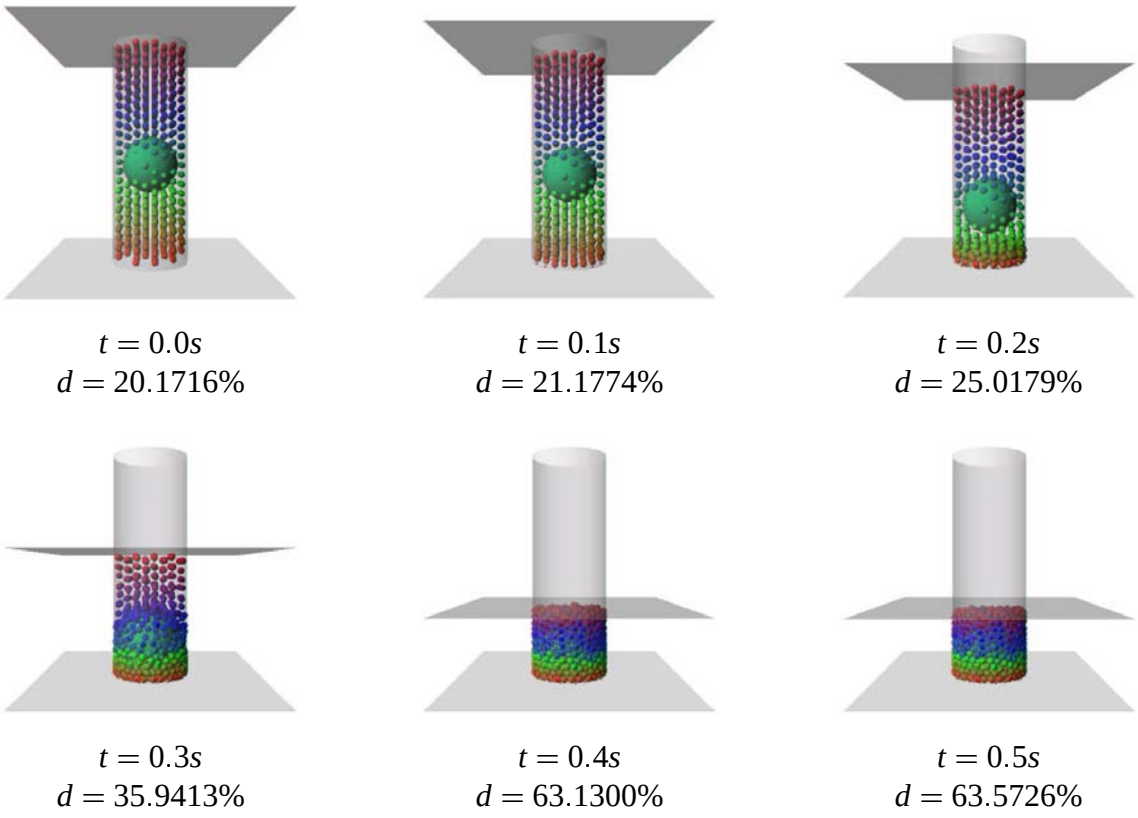


Figure 5.5: The densities for the cases One1, One2, One3 and One4.

grain was not fully covered, leaving a large empty space in the density computation. For cases One1 and One2, the "optimal" densities seem to have been reached earlier, while the larger cases One3 and One4 needed more time but they also reached a higher density.

The initial density is strictly proportional to the total quantity of material used in the experiment. The comparison of One1 and One4 shows that with the same mass of raw material, higher densities are reached with more and smaller grains.

Snapshots at various time intervals for the cases One1 and One3 are given in figures 5.6 and 5.7. In these two cases the proportions of medium and small grains are roughly the same. In One1, the height of the packed media is 31mm, just above the diameter of the large grain, while in One3 it is 53mm, about twice that diameter. The expected higher overall density of the latter case can be explained by noting that large grains tend to reduce the density because of the empty space they generate around them. To fill up that space and further increase the density, grains up to 1000 times smaller might be needed.



$t = 0.0s$
$d = 20.1716\%$

$t = 0.1s$
$d = 21.1774\%$

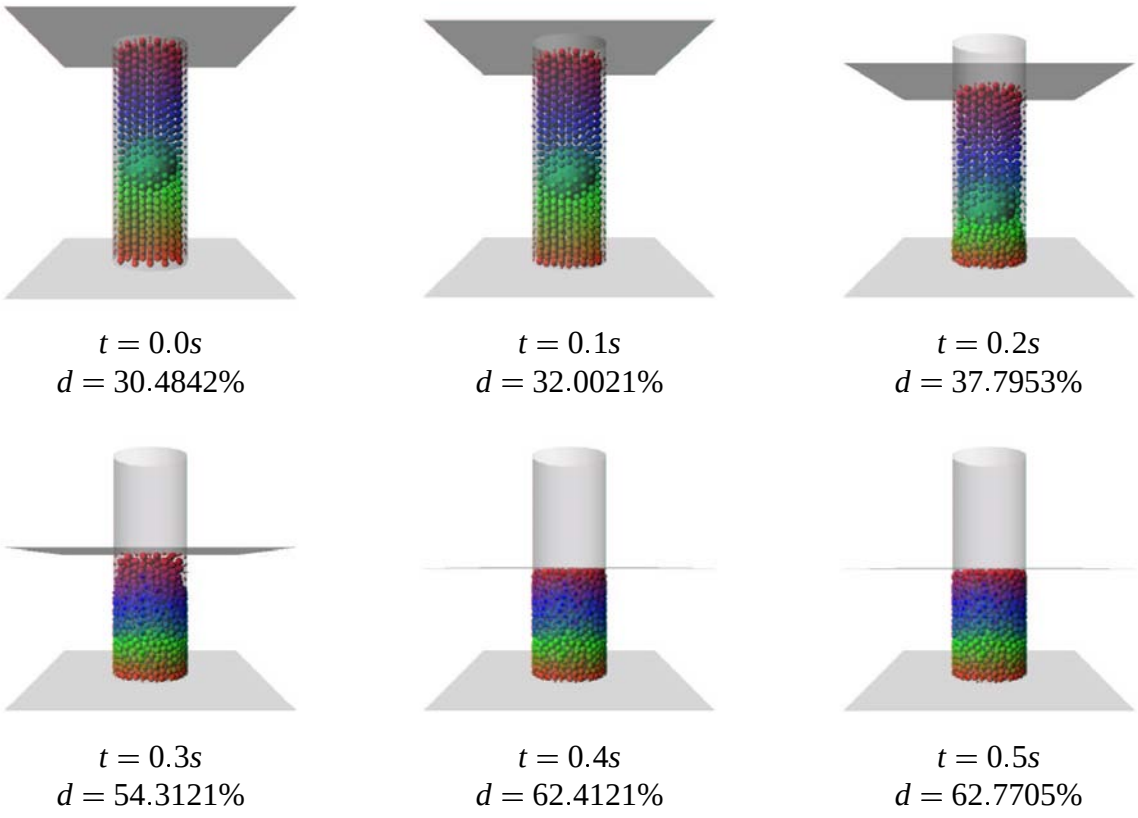
$t = 0.2s$
$d = 25.0179\%$

$t = 0.3s$
$d = 35.9413\%$

$t = 0.4s$
$d = 63.1300\%$

$t = 0.5s$
$d = 63.5726\%$

Figure 5.6: The packing of case One1 at every tenth of second.



$t = 0.0s$
$d = 30.4842\%$

$t = 0.1s$
$d = 32.0021\%$

$t = 0.2s$
$d = 37.7953\%$

$t = 0.3s$
$d = 54.3121\%$

$t = 0.4s$
$d = 62.4121\%$

$t = 0.5s$
$d = 62.7705\%$

Figure 5.7: The packing of case One3 at every tenth of second.

5.4 Selecting the correct vibration

A new set of simulations was performed to study the influence of the vibration frequency on the overall packing process. These simulations involve 5592 grains, all approximately 2mm in diameter. The amplitude of the vibration is set to 2mm. Vibrations are active at 2Hz, 4Hz, 5Hz or 10Hz for 10 seconds, then stop and the packing is left to stabilize for another 2 seconds.

Figure 5.8 shows snapshots of the packings after a few seconds of vibrations. At low frequencies, the packing is dense but there has not been much rearrangement among the grains, as shown by the layered colors. At medium frequencies, the packing remains quite dense, yet with more rearrangements. In these two configurations, the optimal face-centered cubic packing is clearly visible in several places. Finally, at high frequencies, the packing suffers from too much shaking: the time between two consecutive peaks of the tapping cycle is too short for the grains to stabilize, introducing wave effects, and yielding a lower average density. However, this behavior could be interesting in a preliminary phase where the goal would be to guarantee a good spatial distribution of the various grain types, and denser packing would be attained in a second phase with a lower frequency. Applying such a technique to grains of different sizes requires special attention to be paid to undesired segregation effects.

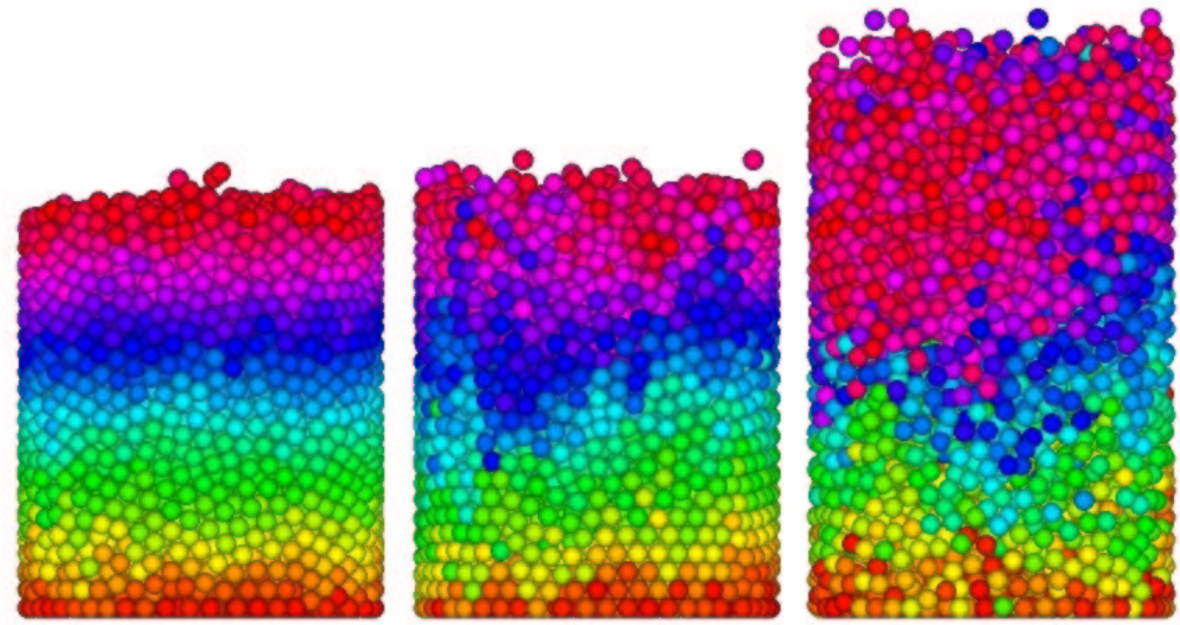


Figure 5.8: Snapshots at various frequencies: 2Hz (left), 5Hz (center) and 10Hz (right).

The above deductions obtained by simple visualization are fully confirmed by the detailed measurements taken during the simulation. Figures 5.10, 5.11, 5.12 and 5.13 on pages 84 to 87 show for each frequency the average density of the packing, a mobile average of said density, the maximal velocity attained by a single grain, and 20 local densities measured by horizontal layers. The dashed vertical grid-lines indicate the beginning of the tapping cycles.

All densities are expressed here as the percentage of volume occupied by the spheres. For reference, the theoretical maximum is approx. 74% (for an infinitely large cylinder), while

values obtained in practice range between 60% and 66%.

Due to the way it is computed, the average density is very chaotic during the vibrations. However, as soon as the vibrations stop, its value increases and stabilizes at a peak. At 10Hz, no special behavior was observed. At 5Hz on the other hand, short series of local peaks and little variations – just reflecting the movement of the vibrating plate – are separated by larger gaps. This is clearly an indication that important rearrangements occur from time to time. At 4Hz the curve is smoother, but a larger rearrangement takes place at 8.75s. The maximal density is reached by the 2Hz simulation.

The moving average of the density is computed as the (unweighted) average over the last second. At 10Hz it is relatively low, in the 0.45 to 0.55 range, while at 5Hz, 4Hz and 2Hz it is not only higher, in the 0.55 to 0.6 range, but also shows a tendency to increase.

The maximal velocity among the grains is mainly used to detect the moment of complete settlement of the packed media. At 5Hz, it takes place around half a second after the last vibration, whereas at 10Hz more than a second is required. The rearrangement occurring at 8.75s in the 4Hz case is clearly visible.

The local densities measured by horizontal layers also help understand the behavior of the media. At the top, the values obviously decrease and remain zero, this is a side effect of the way the packing is initiated. At 10Hz, the middle part is very chaotic while the bottom part seems more stable. About once per second, a relatively large number of grains are propelled at quite some height, yielding non-zero densities in the high-middle low-top area. At 5Hz the overall behavior seems less chaotic. The variation in the lowest layer is transmitted to the top through the dense media. The 4Hz case is similar, up to the few grains ejected upwards at 8.75s. Finally, at 2Hz, the time between two tapping cycles is sufficient for the whole assembly to stabilize, yielding a sequence of independent yet similar density changes.

These results tend to prove that a medium frequency of 4Hz to 5Hz gives better results than a high one of 10Hz. A low frequency such as 2Hz does not induce enough reorganization in the assembly, therefore the final quality of the packing depends too much on the initial state before the vibration. This is confirmed by the final value of the density shown in table 5.2:

Frequency	**10Hz**	**5Hz**	**4Hz**	**2Hz**
Density	59.92%	60.40%	60.61%	60.60%

Table 5.2: Final densities for various vibration frequencies.

These values can be compared with an artificial value of 63.32% obtained by intersecting an infinite optimal face-centered cubic packing and the test cylinder, as shown in figure 5.9. It is also interesting to compare them with the values obtained with many more smaller grains as shown in section 5.8.

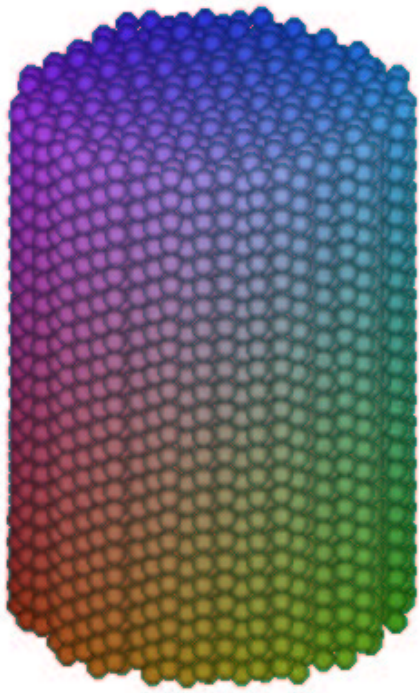


Figure 5.9: Artificially filling the cylinder with a face-centered, theoretically optimal packing.

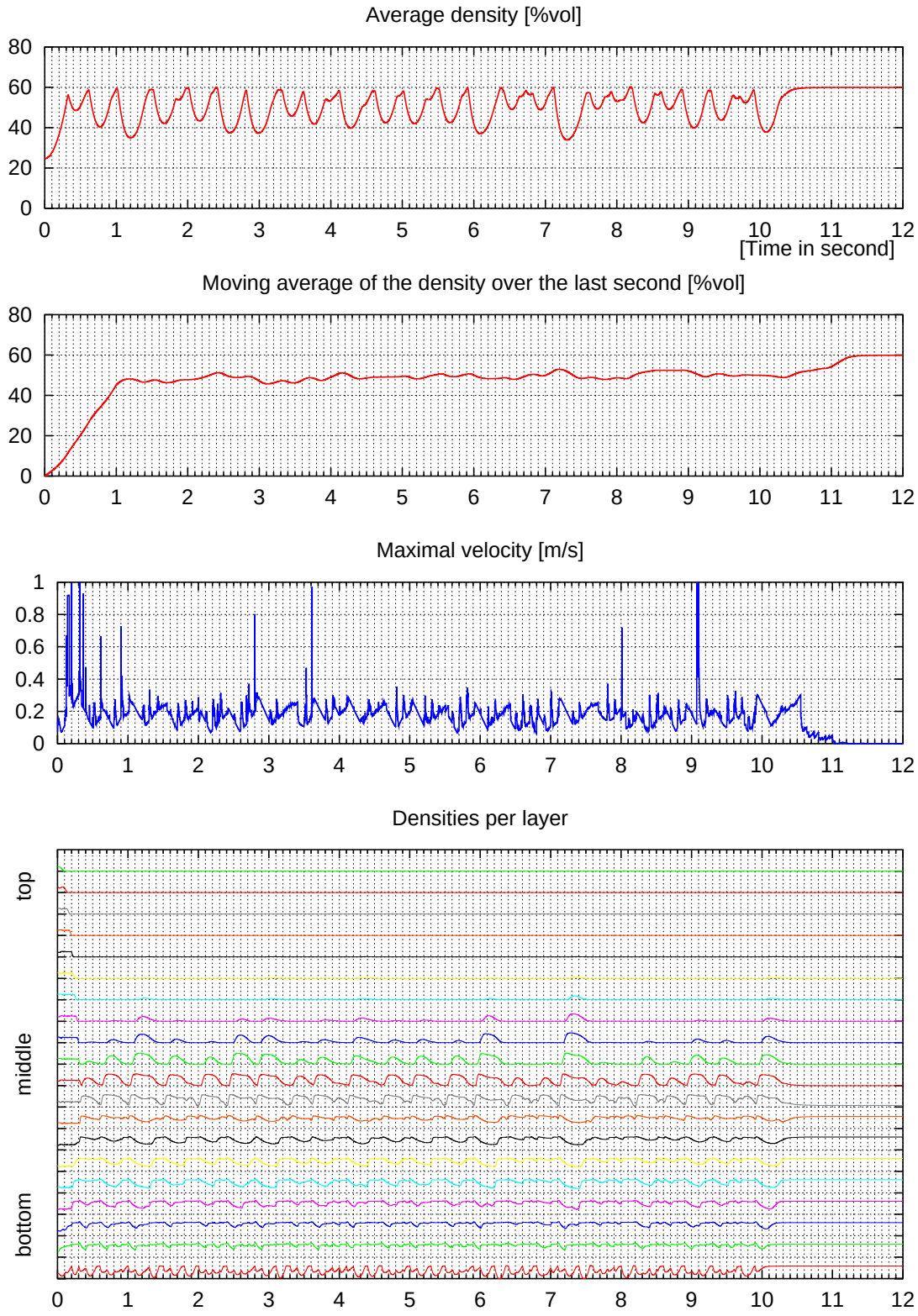


Figure 5.10: Measures computed at 10Hz.

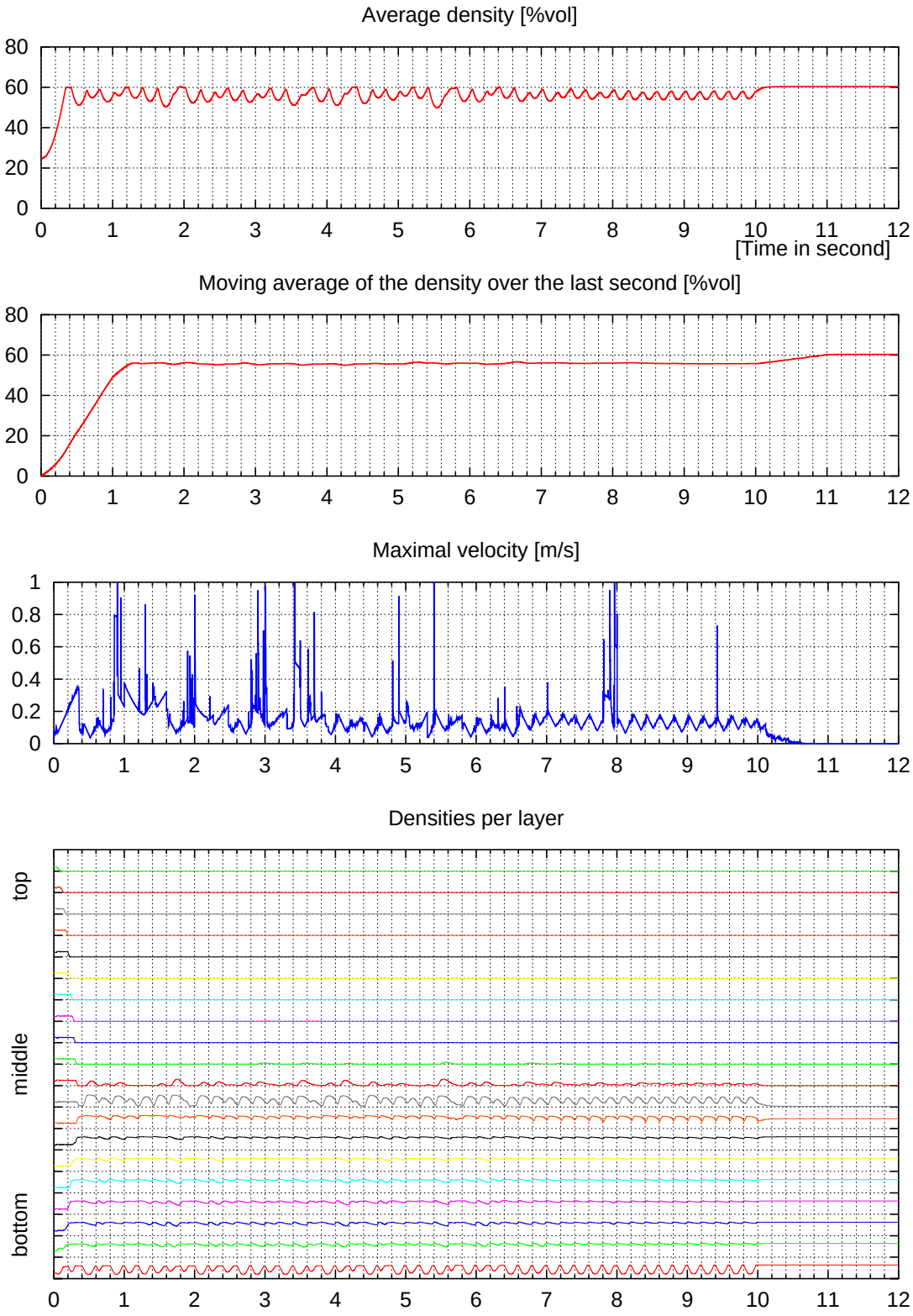


Figure 5.11: Measures computed at 5Hz.

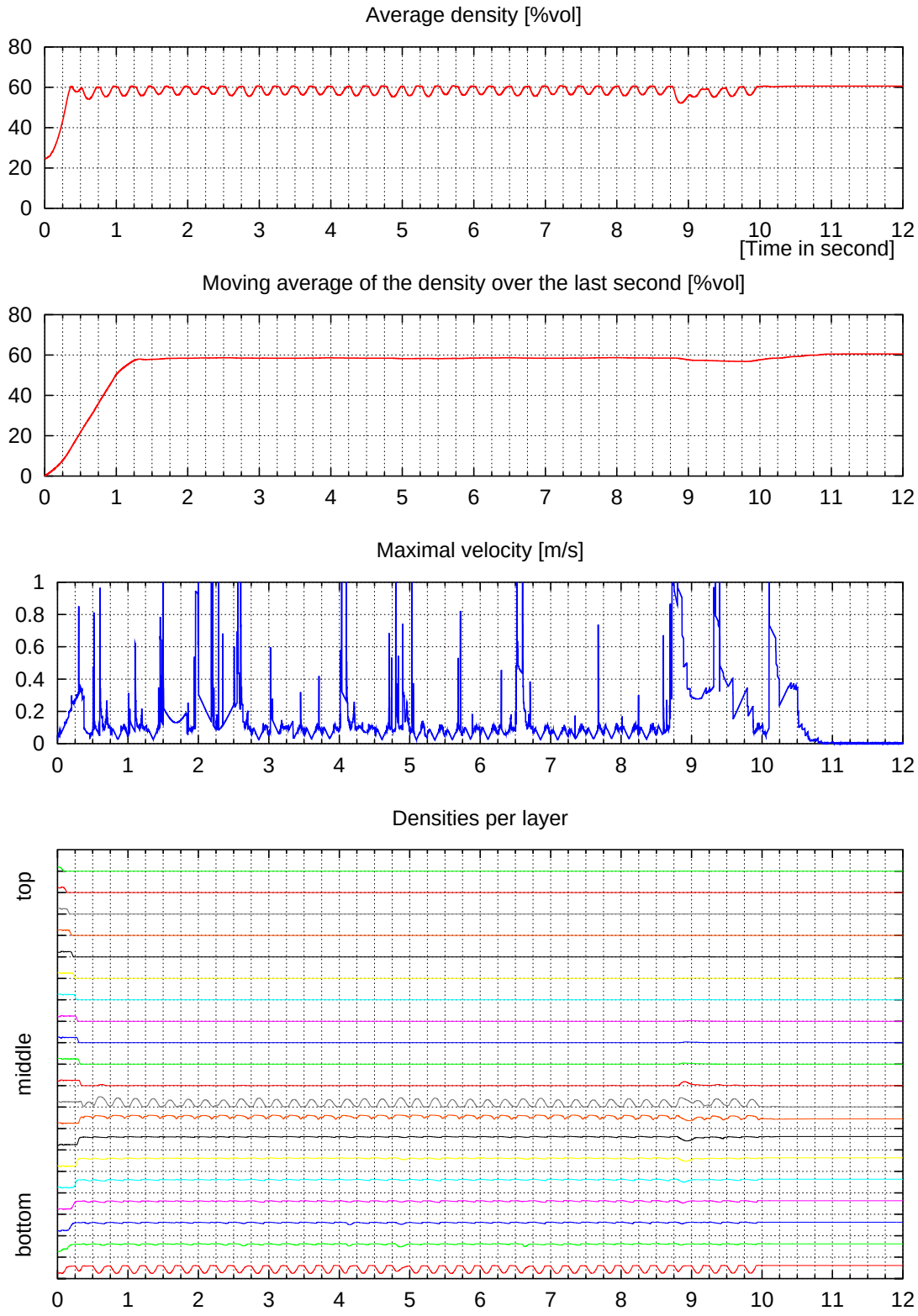


Figure 5.12: Measures computed at 4Hz.

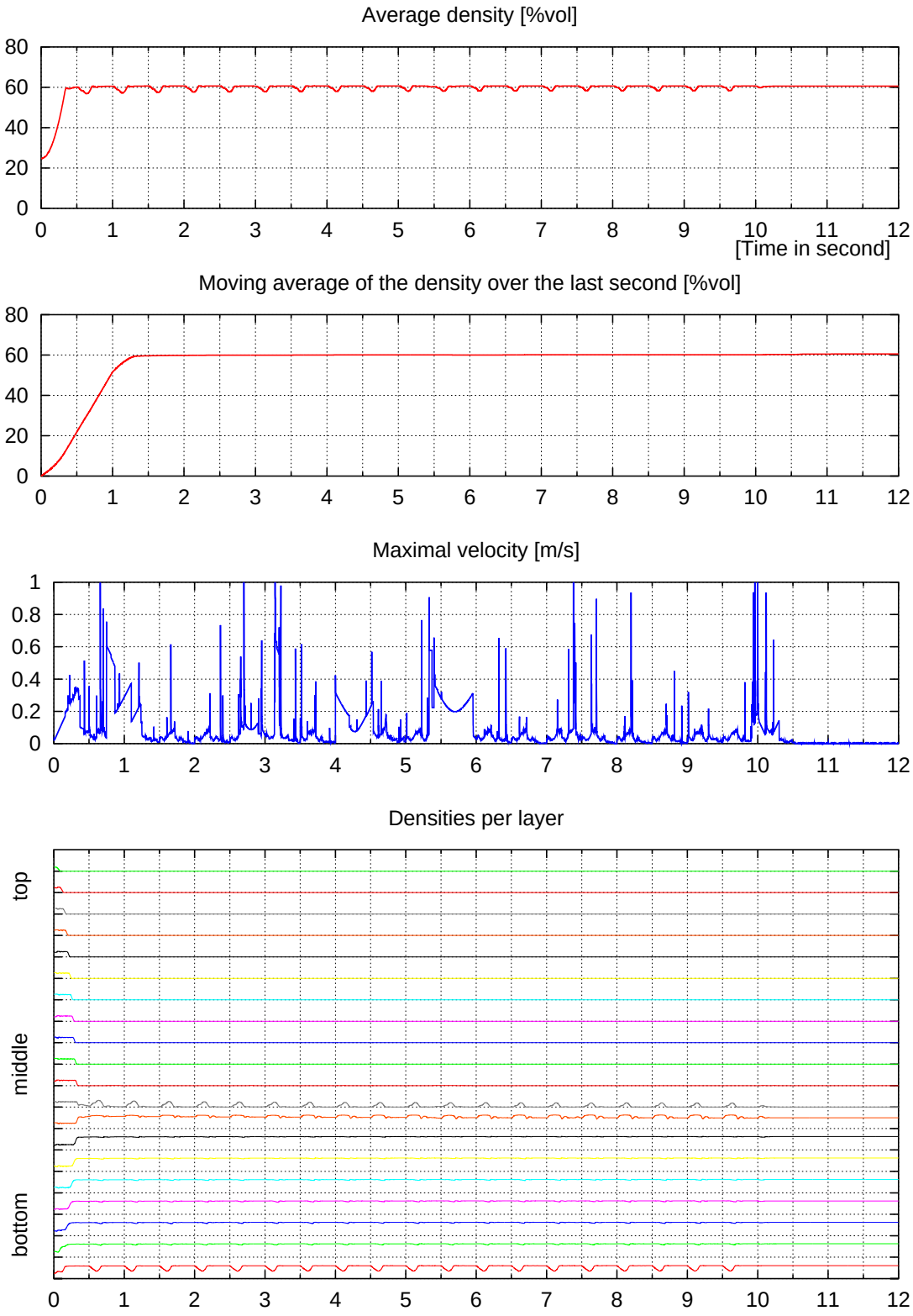


Figure 5.13: Measures computed at 2Hz.

5.5 Filling the space between the large grains

When considering grains with size ratio 1:5:35 and mass ratio 70:20:10, there must be more than 6000 small grains for each large grain. To study the behavior of medium and small grains near the large ones, we have set up some cases where the large grains are fixed and truncated, thus only considering a small portion of the space comprised between large grains.

These simulations give a qualitative insight of how the spaces between the large grains are filled, but no quantitative results that could be extended to the global packing. In particular, no density measures were made due to the strong bias induced by the truncated geometry.

The large pseudo-spheres have a diameter of 35mm, and are placed on the vertices of a face-centered cubic grid. The mesh size of this grid was chosen between 35mm (spheres are touching) and 40mm. The medium and small spheres have a diameter of 5mm and 1mm.

Two configurations were tested. In the first one, a section was cut by 6 vertical planes forming a hexagon. In the second, a section was cut by a vertical cylinder. Figure 5.14 shows vertical projections of these geometries.

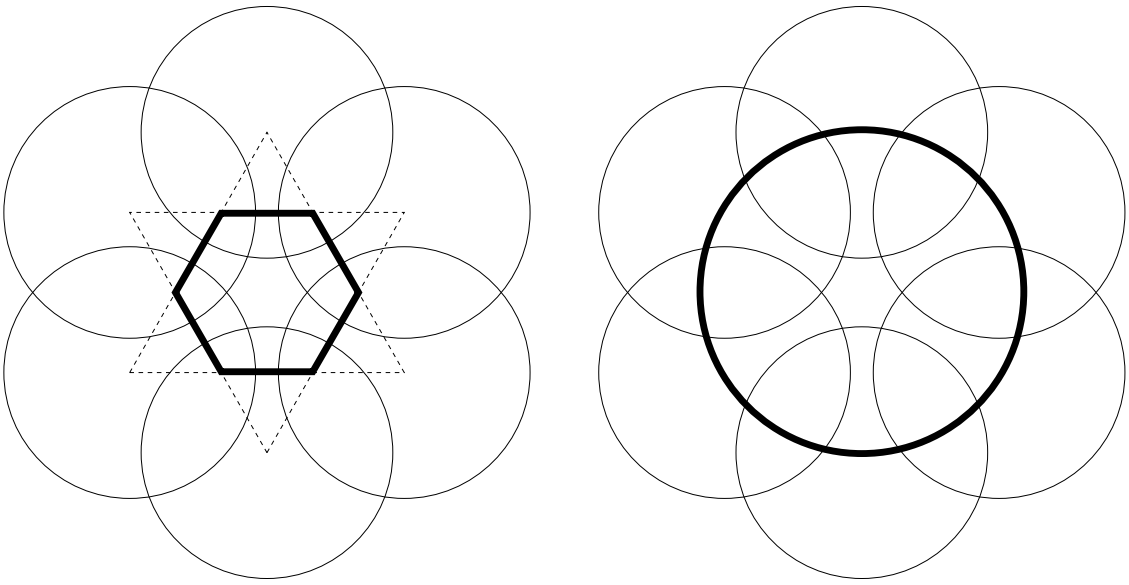
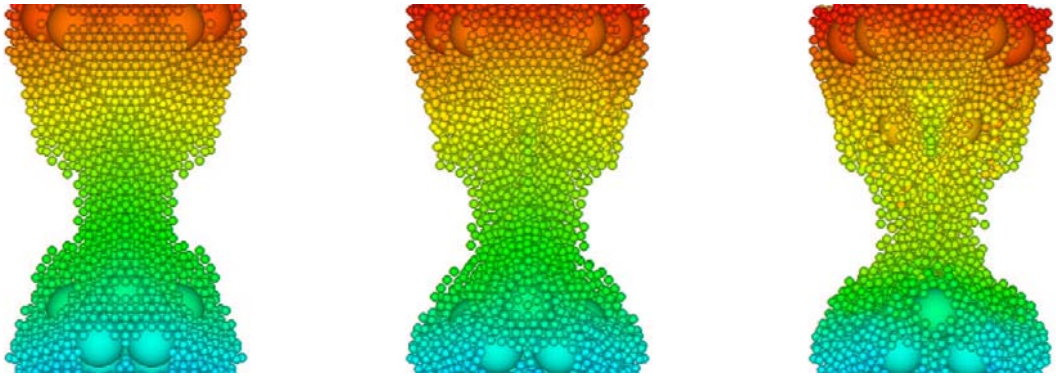
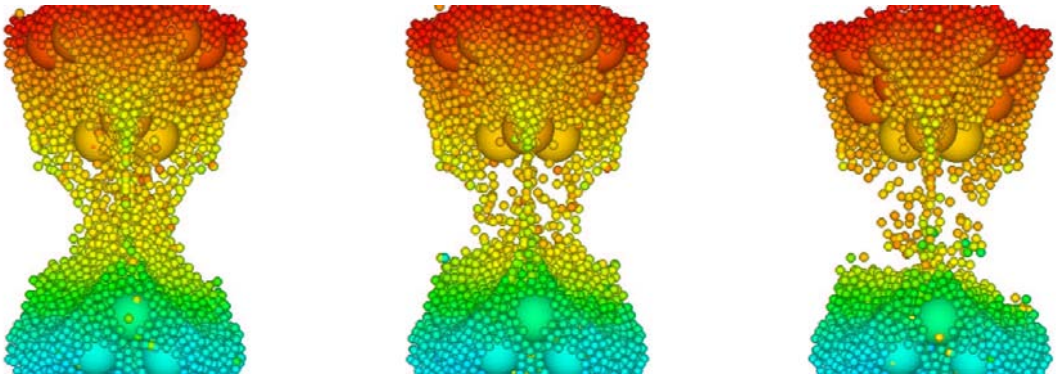


Figure 5.14: Hexagonal and cylindrical sections cutouts, the 6 discs represent two layers of large grains disposed in a loose face-centered cubic packing

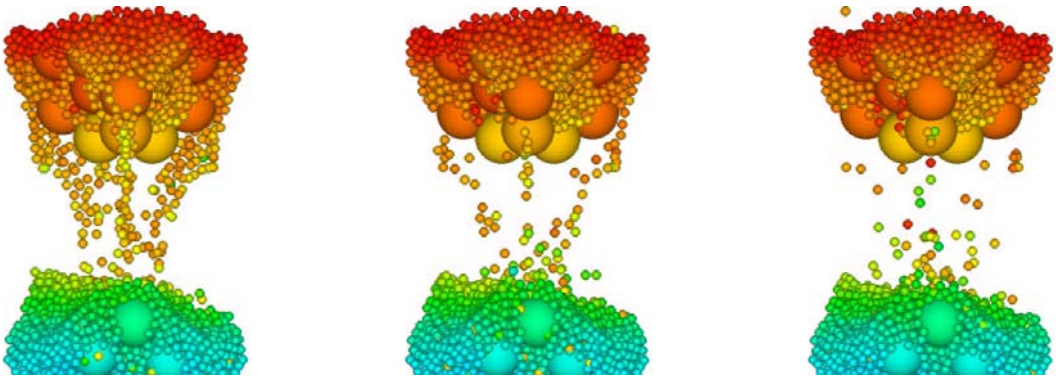
Figure 5.15 shows 9 snapshots at various time intervals of an hexagonal section, with the large grains at distance 35mm (ie touching). The simulation comprised 33 medium and 5929 small grains. Figure 5.16 shows 9 snapshots at various time intervals of a cylindrical section, with the large grains at distance 40mm. The simulation comprised 178 medium and 24360 small grains.



$t = 0.00s$ $t = 0.01s$ $t = 0.02s$

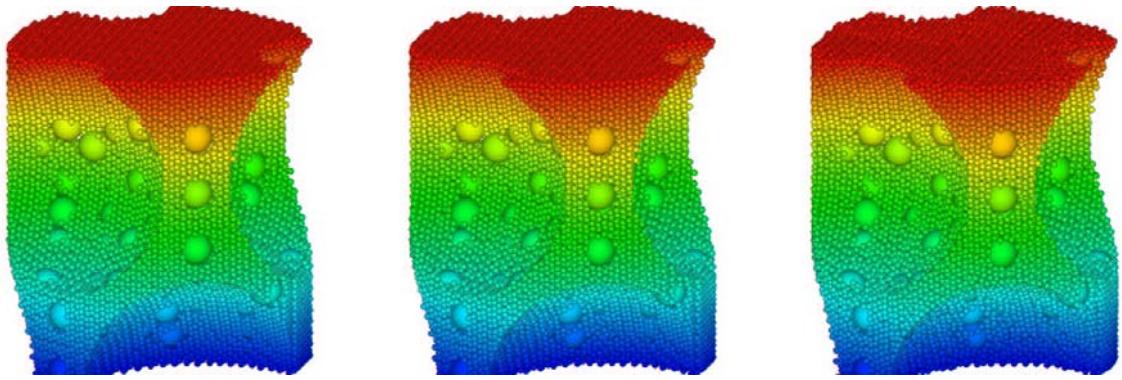


$t = 0.03s$ $t = 0.04s$ $t = 0.08s$

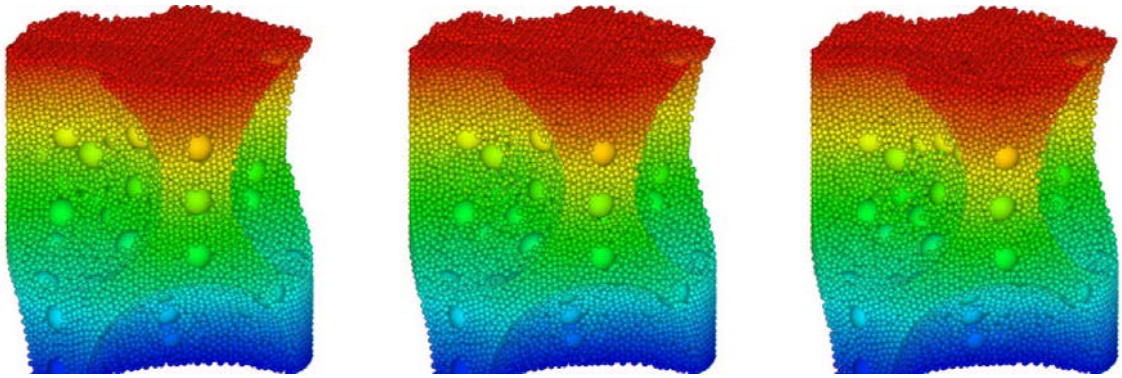


$t = 0.12s$ $t = 0.16s$ $t = 0.20s$

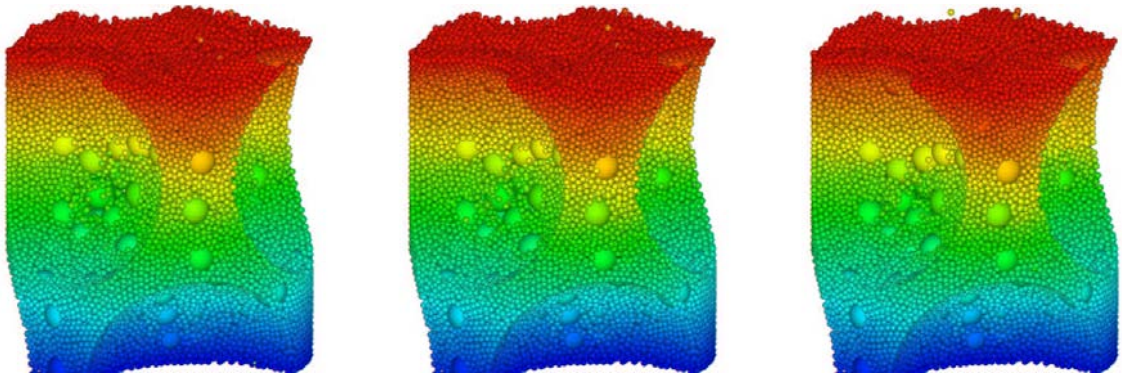
Figure 5.15: First example of interstitial view, based on an hexagonal cut. Close-up of the interstice between three touching large grains.



$t = 0.02s$ $t = 0.04s$ $t = 0.06s$



$t = 0.08s$ $t = 0.10s$ $t = 0.12s$



$t = 0.14s$ $t = 0.16s$ $t = 0.18s$

Figure 5.16: Second example of interstitial view, based on a cylindrical cut. Global view of an area truncated by 3 large grains and 3 half large grains.

5.6 Covering one large grain

Another set of simulations concerns the rearrangement of medium and small grains around the large ones. We have selected in this case one large grain whose diameter is equal to the inner diameter of the cylinder. Medium and small grains are then stacked above the large one and the whole assembly is vibrated.

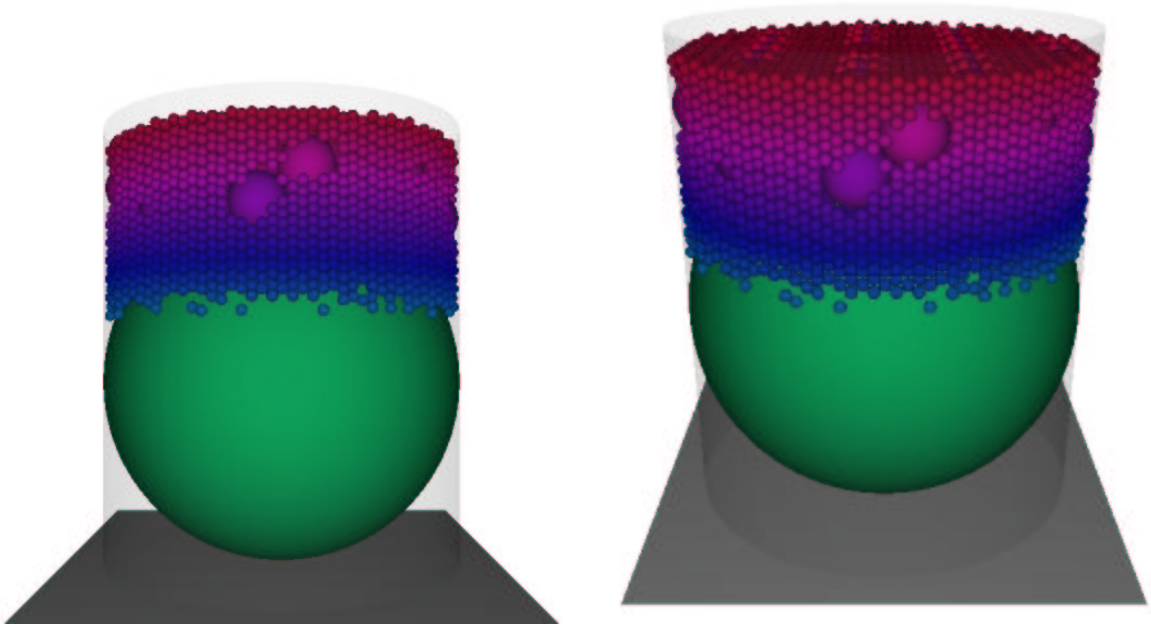


Figure 5.17: Covering a large grain that fills the cylinder, initial situation.

Figures 5.18 and 5.19 give snapshots of the interesting layers during the first second of the simulation. One can notice the rearrangement of the small grains along the border of the cylinder.

Density measures were taken, but the dominance of the large grain introduces a strong bias, even in the layered densities. They are given in figure 5.20 but should be considered with extreme care. Aside from the usual average density, the moving average of said density and the layered densities, we have added the structure of the layered densities at every tenth of a second during the first second. As expected, layers 1 to 11 are dominated by the large grain and reflect the variation due to the vibrations, while layers 12 to 18 see a mixture of all grains. The density decrease in this part is due to the increase in the surface of the large grains in each layer, which itself yields a loss similar to the boundary effect near the border of the cylinder.

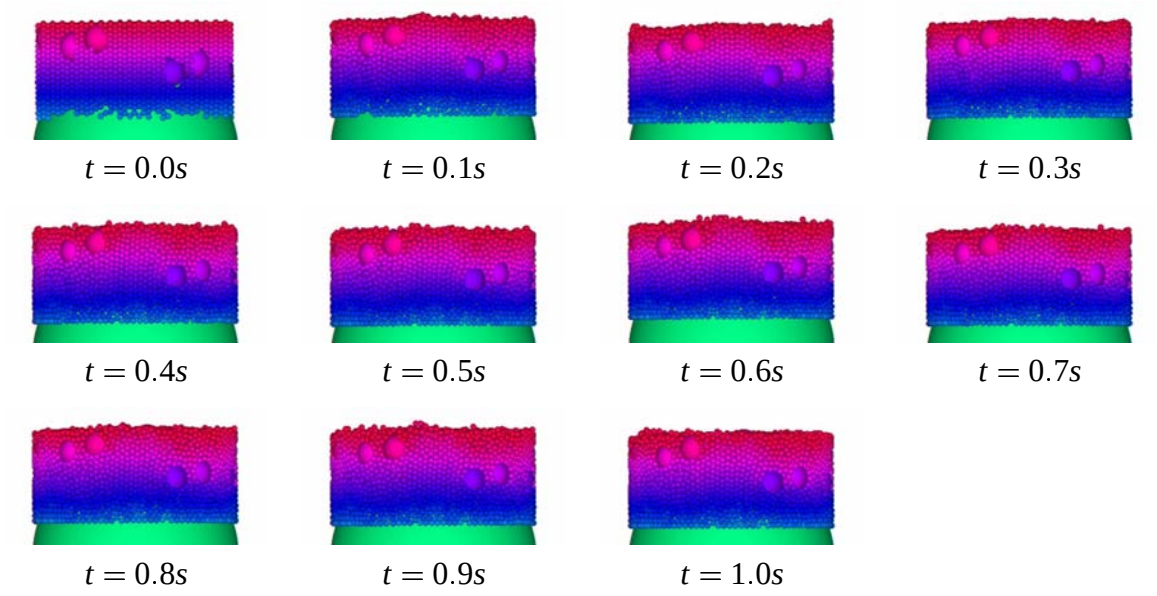


Figure 5.18: Side view of medium and small grains covering one large grain in a narrow cylinder at various time intervals.

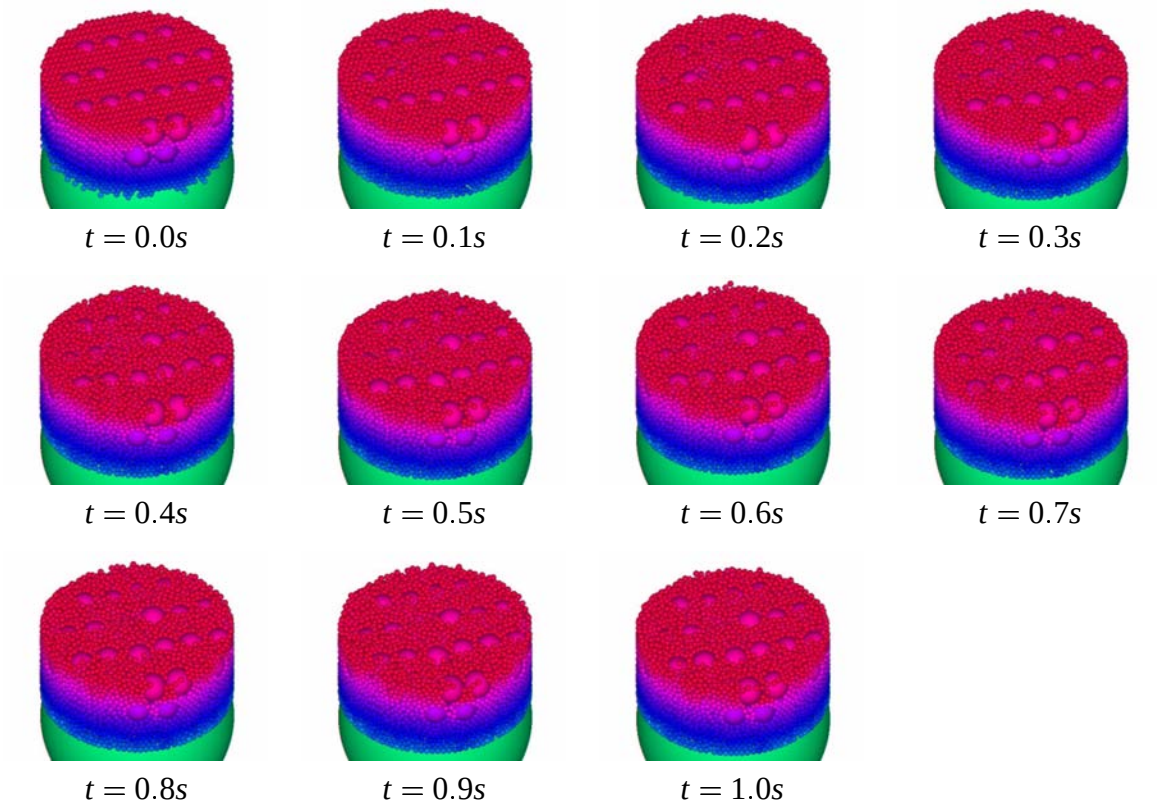


Figure 5.19: Side view of medium and small grains covering one large grain in a narrow cylinder at various time intervals.

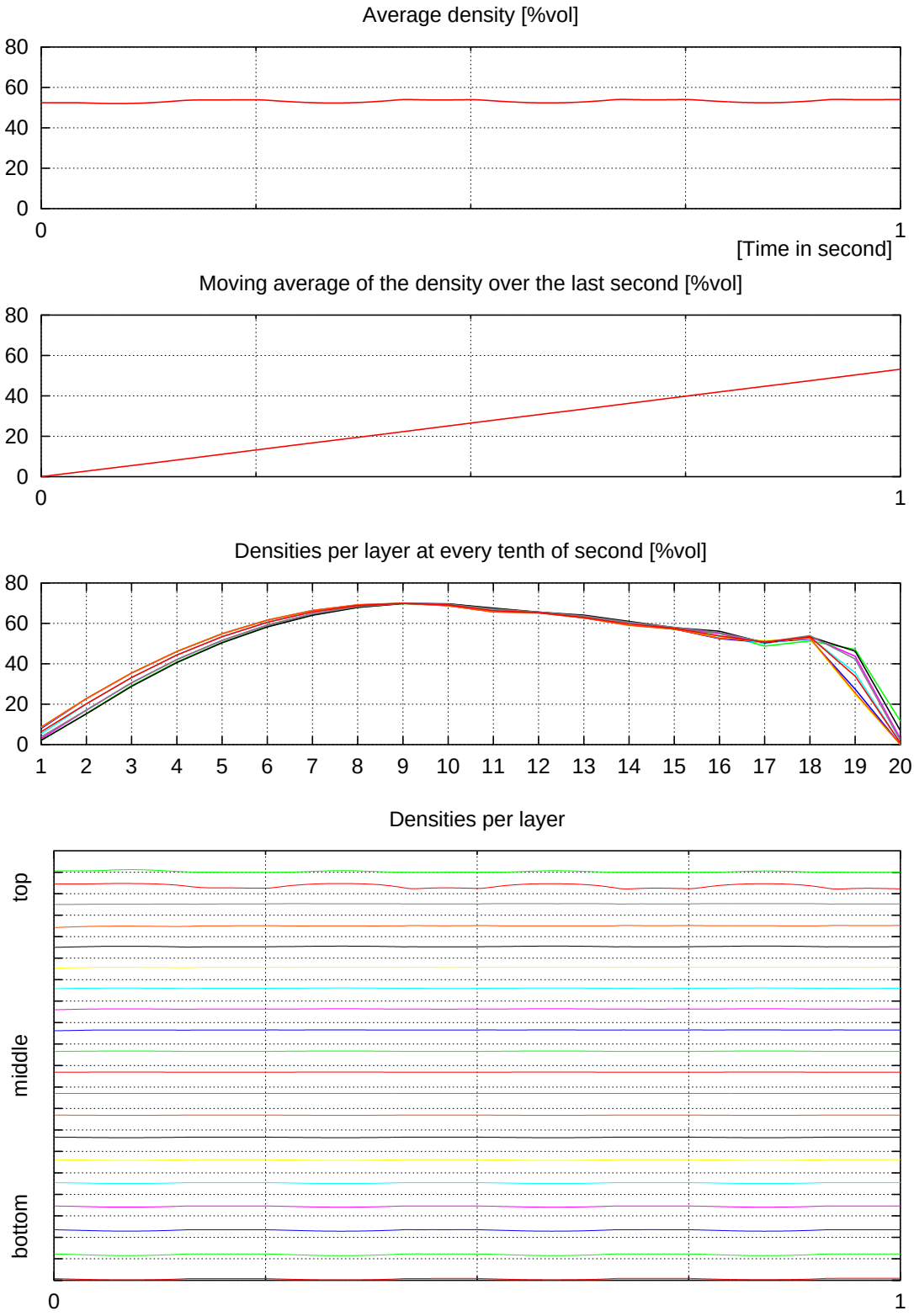


Figure 5.20: Density measures for a half-covered large grain.

5.7 Other simulations

More experiments were performed with one or two large grains. Figure 5.21 shows the result of selecting the number of medium and small grains according to a 70% – 20% – 10% mass distribution. With grain sizes of 35mm, 5mm and 1mm this yields 1 large, 98 medium and 6125 small grains, see table 5.3.

		large	**medium**	**small**
grain diameter	[mm]	35	5	1
volume of one grain	[mm^3]	22'450	65.45	0.5236
relative mass	[%]	70	20	10
number of grains (fig 5.21)		1	98	6125
number of grains (fig 5.22)		2	196	12250

Table 5.3: Number of grains for the first try.

However, with only one large grain it is not possible to enforce this distribution: either the cylinder is too large, and there is not enough material to cover the large grain as shown in figure 5.21, or the cylinder is too narrow and the large grain will block the medium ones, as shown in figure 5.22. In this case, the layered densities show very large variations, as shown in figure 5.23. In the third graph, the green line corresponds to 0.1s, the blue line to 0.2s. After 0.2s the simulation was stopped.

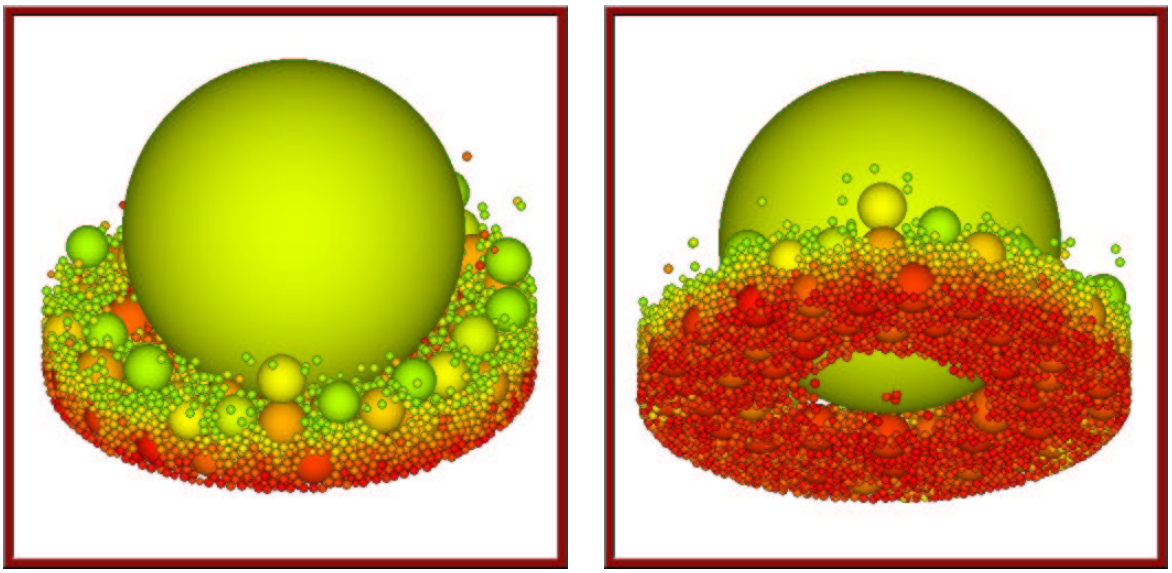


Figure 5.21: Two views of the simulation with a large cylinder: the large grain is not covered.

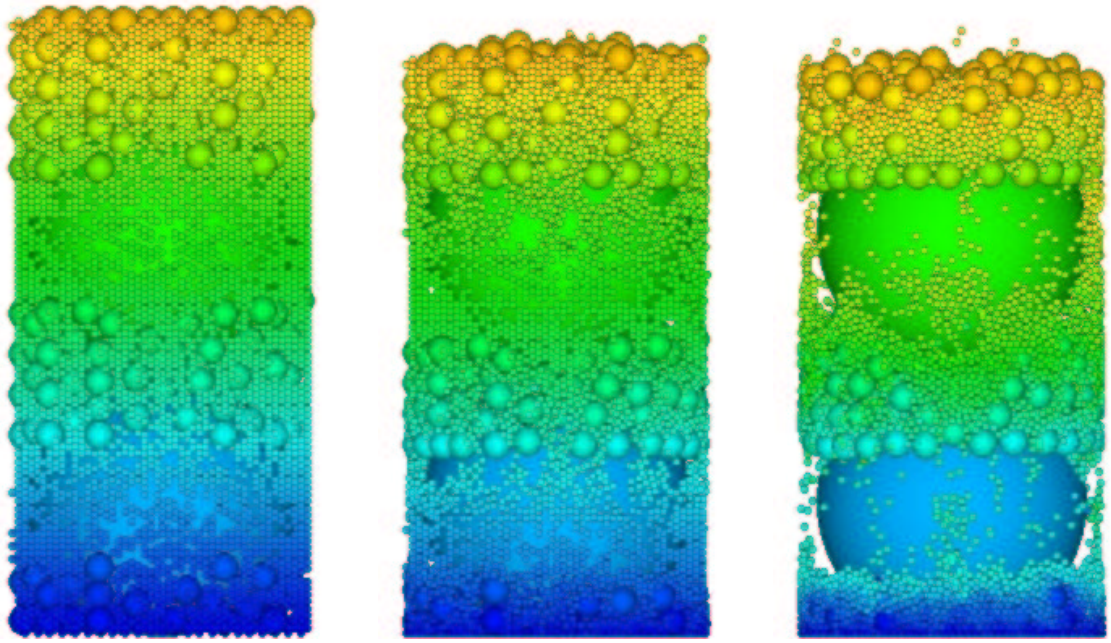


Figure 5.22: Three views of a simulation with a small cylinder, introducing strong segregation among the grains.

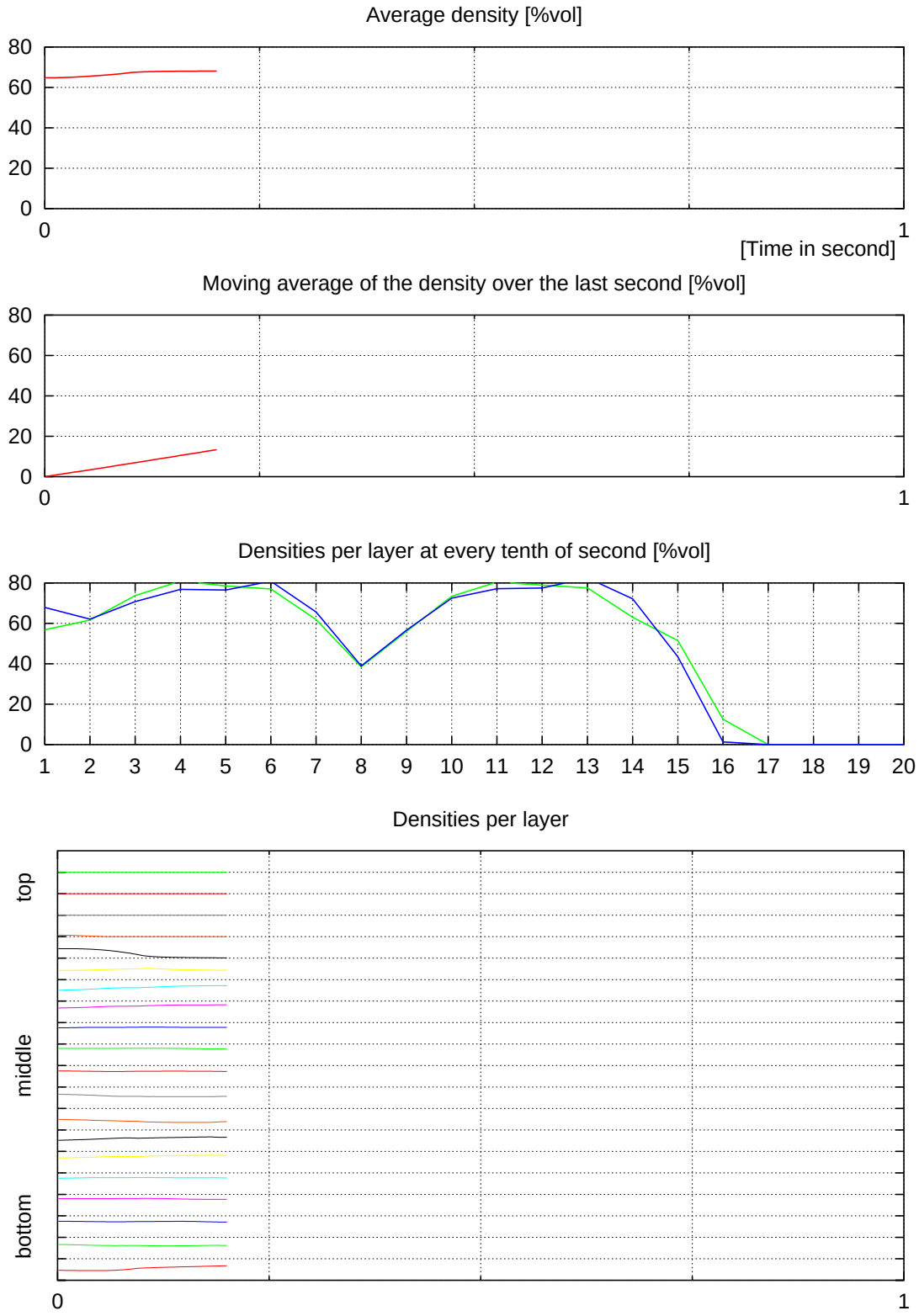
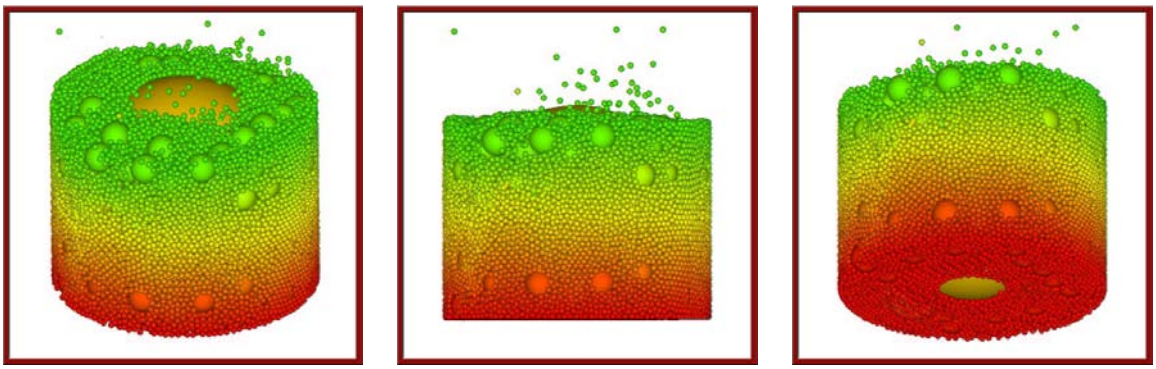


Figure 5.23: Densities measures when medium grains are blocked.

The simulation was then repeated with more material, see table 5.4 and figure 5.24. This time the large grain is almost covered, and density measures can be taken. They are shown in figure 5.25. Since only one grain of 35mm was used in the 10cm high cylinder, only the bottom layers give interesting results. No vibrations were used in this short simulation. The abnormally high density achieved by the first layer is due to the penetration of the large grain in the floor — permitted by the contact model — which was not accounted for in the layered density computation.

		large	**medium**	**small**
grain diameter	[mm]	35	5	1
volume of one grain	[mm^3]	22'450	65.45	0.5236
relative mass	[%]	55.4	12.3	32.3
number of grains		1	76	24971

Table 5.4: Number of grains for the second try.



Top view Middle view Bottom view

Figure 5.24: Three views of the simulation with the grains of table 5.4: the large grain is almost covered.

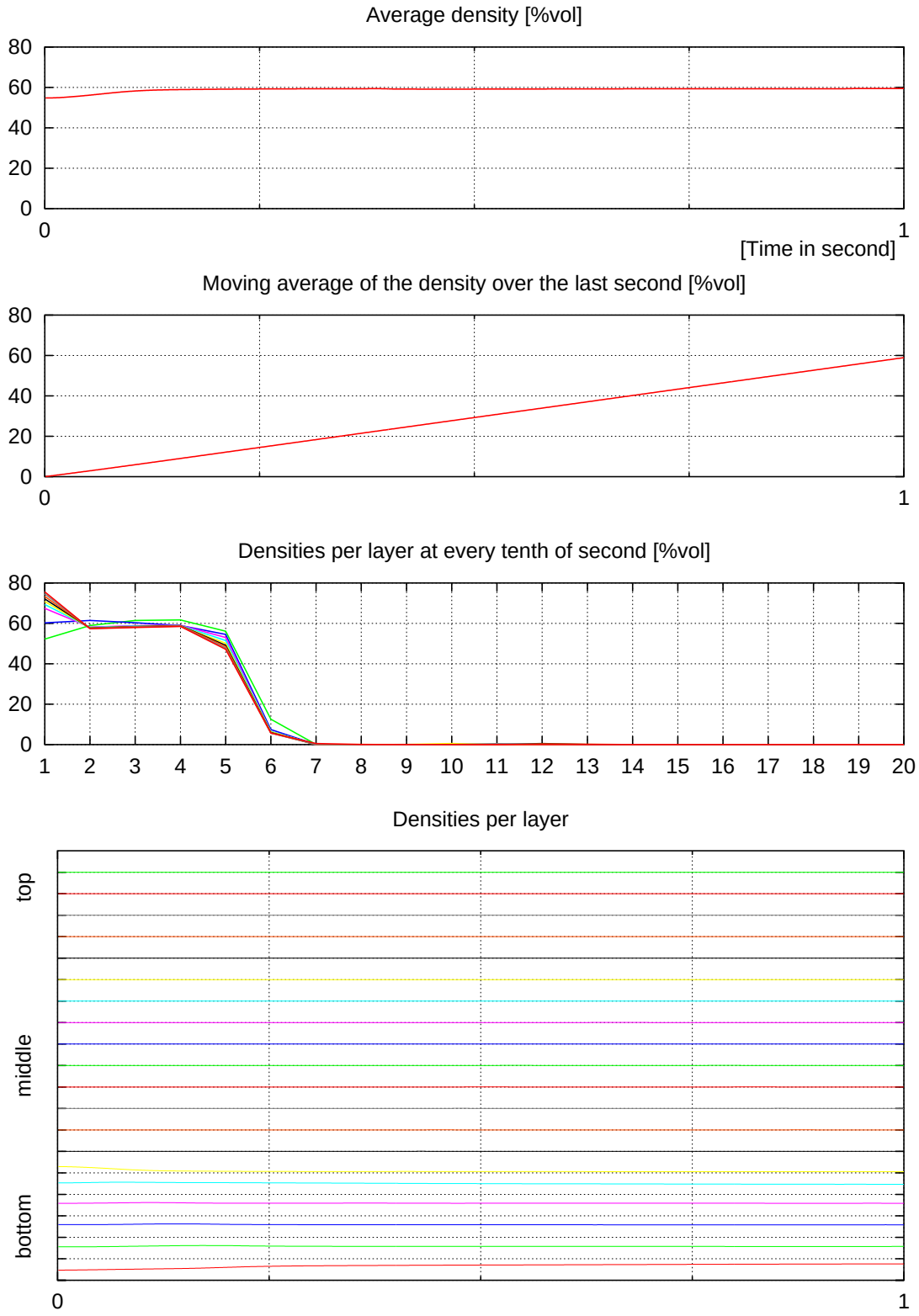


Figure 5.25: Densities measured for the second try.

5.8 Experimental validation of the simulation

Even if the results obtained by simulation seem reasonable, it is necessary to further assess the validity of the method and of the computer program by comparing simulated values with a similar real experiment. As mentioned before, the powders currently used at the Energetic Material Labs are much too fine for the simulation. With particle sizes in the 10^{-6}m to 10^{-3}m range, filling up a 35mm by 100mm cylinder requires at the very least several million grains, which is currently out of reach.

The validation effort has thus been directed towards larger spheres with diameters between 1mm and 25mm, in cylinders of 30mm and 36mm. The spheres are made of hard steel and have a density of 7.78 g/cm^3. Figure 5.26 shows the complete experimental setting used by Folly (2000). Three categories of experiments were conducted.

Figure 5.26: The experimental setup as used at the Energetic Material Labs of Swiss Defense Procurement Agency at Thun.

5.8.1 Unimodal case

A first case involves only spheres of 1mm of diameter, in an experiment similar to the simulations we performed to select the best frequency, as discussed in section 5.4. Approximately 26'000 spheres of diameter 1mm are poured into the cylinder and vibrated. The resulting assembly is shown in figure 5.28. In the 30mm cylinder, the density reached 66%, while in the 36mm cylinder it was 64%. Several simulations were conducted with approximately 13'000 spheres in a 30mm cylinder and produced packings with densities between 63.8% and 65.1%. Various pictures are shown in figure 5.27.

Visual comparison of the experiment and the simulation indicates a relatively good match, despite the fact that the simulation only lasted two seconds. However, the various simulations showed a strong dependence of the resulting packing to the initial placement of the grains. A possible way to overcome this is to start with an empty cylinder and pour grains one by one from the top. Such a setup has already been tested on small cases and should be tried in future simulations.

The simulations reached slightly lower densities than the experiment. We believe this is because fewer grains were used (13'000 vs. 26'000) and again a shorter duration, in other words fewer vibration cycles.

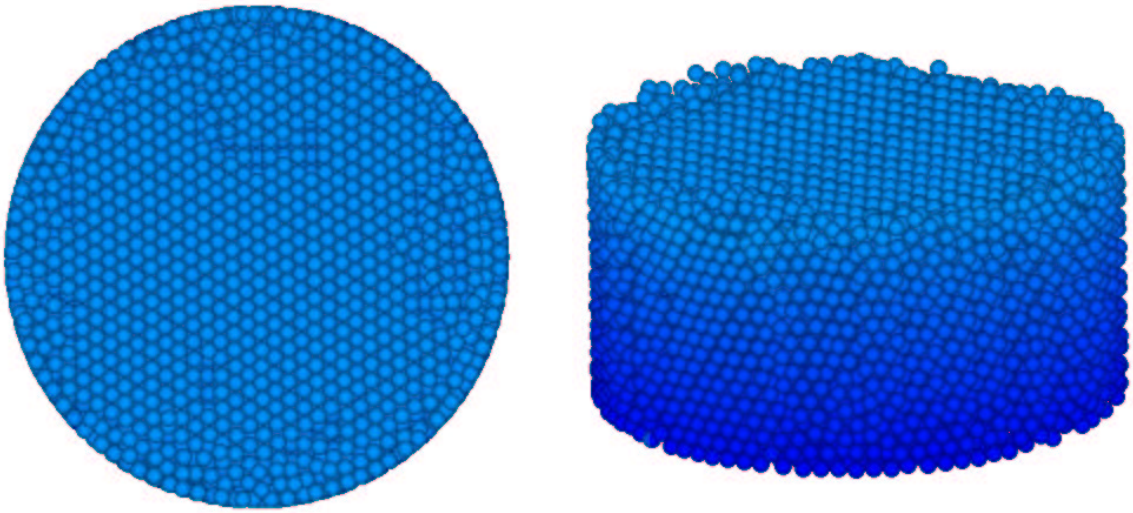


Figure 5.27: Top and side view of the simulation with 13'000 spheres of diameter 1mm in a 30mm cylinder.

5.8.2 Narrow distribution

A second set of experiments involved spheres of three different sizes with a relatively narrow distribution: 4mm, 7mm and 15mm. Two cylinders were used: 30mm and 36mm, see figure 5.29. Both cases used the same set of 4 large spheres (15mm), 56 medium (7mm) and 293 small (4mm). Five repetitions with the 30mm cylinder always gave the same height of

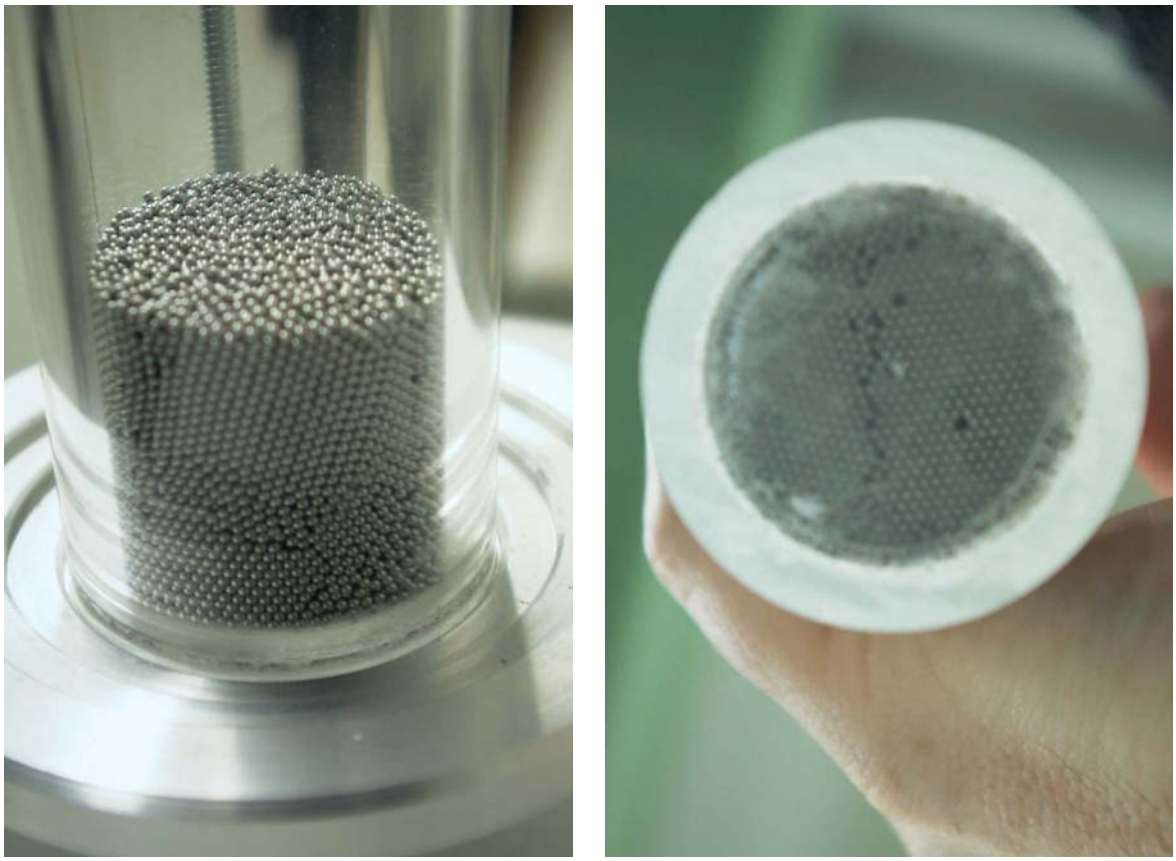


Figure 5.28: Two pictures of the experiment with approximately 26'000 spheres of diameter 1mm in a 30mm cylinder.

62mm, thus a density of 61%. Three repetitions with the 36mm cylinder gave heights of 44mm, 44.5mm and 45mm. Averaged to 44.5mm, this gives a density of 60%.

The simulation produced very similar results. Figures 5.30, 5.31 and 5.32 show the initial grain population as well as packing after 1 and 10 seconds for three cylinder sizes: 30mm, 36mm and 40mm. Figures 5.33, 5.34 and 5.35 show the detailed densities for the three simulations. Table 5.5 lists the grains used and table 5.6 gives a summary of the key results.

		large	**medium**	**small**
grain diameter	[mm]	15	7	4
volume of one grain	[mm^3]	1767	179.6	33.51
relative mass	[%]	26.3	37.3	36.4
number of grains		4	56	293

Table 5.5: Grains used in the narrow distribution.

		30mm	**36mm**	**40mm**
simulated density at 1s	[%]	59.44	58.42	61.49
simulated density at 10s	[%]	60.78	58.84	61.73
experimental density at 50s	[%]	61	60	—

Table 5.6: Densities for the narrow distribution.

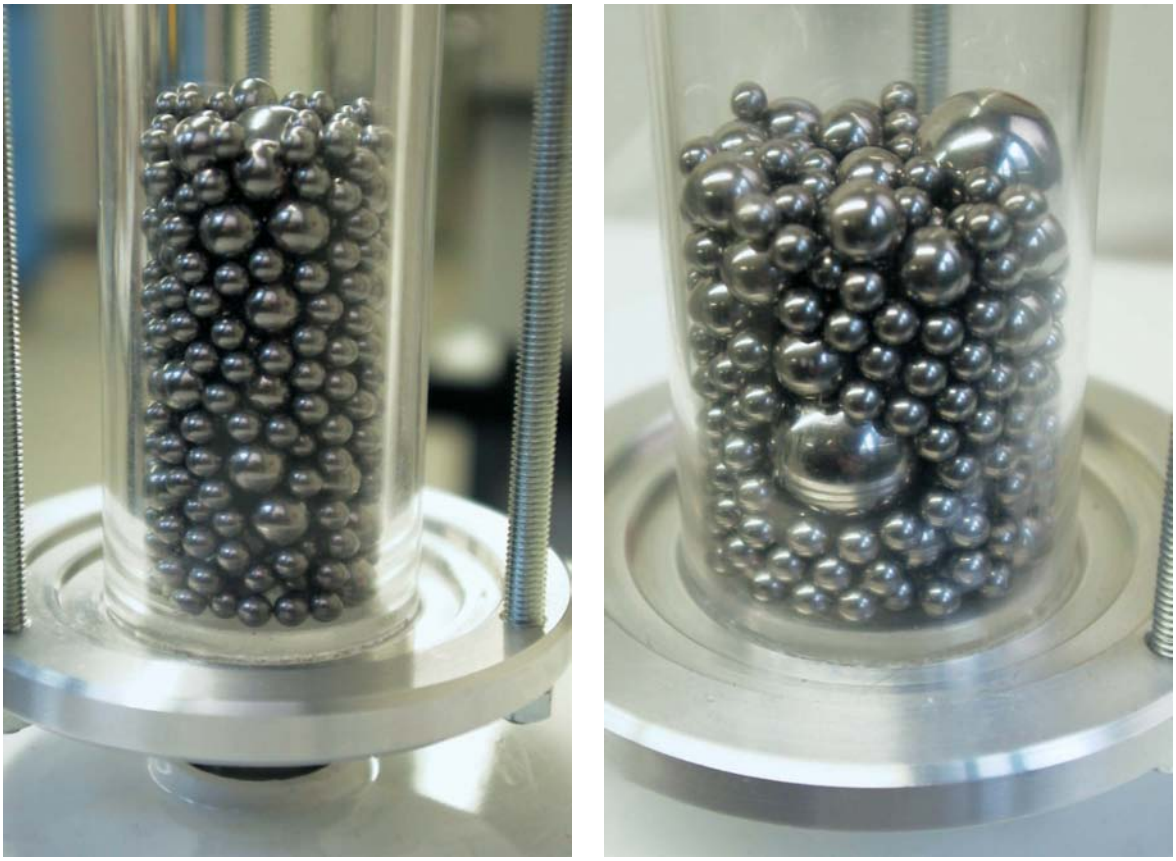


Figure 5.29: Spheres of 4mm, 7mm and 15mm in a 30mm cylinder (left) and a 36mm cylinder (right).

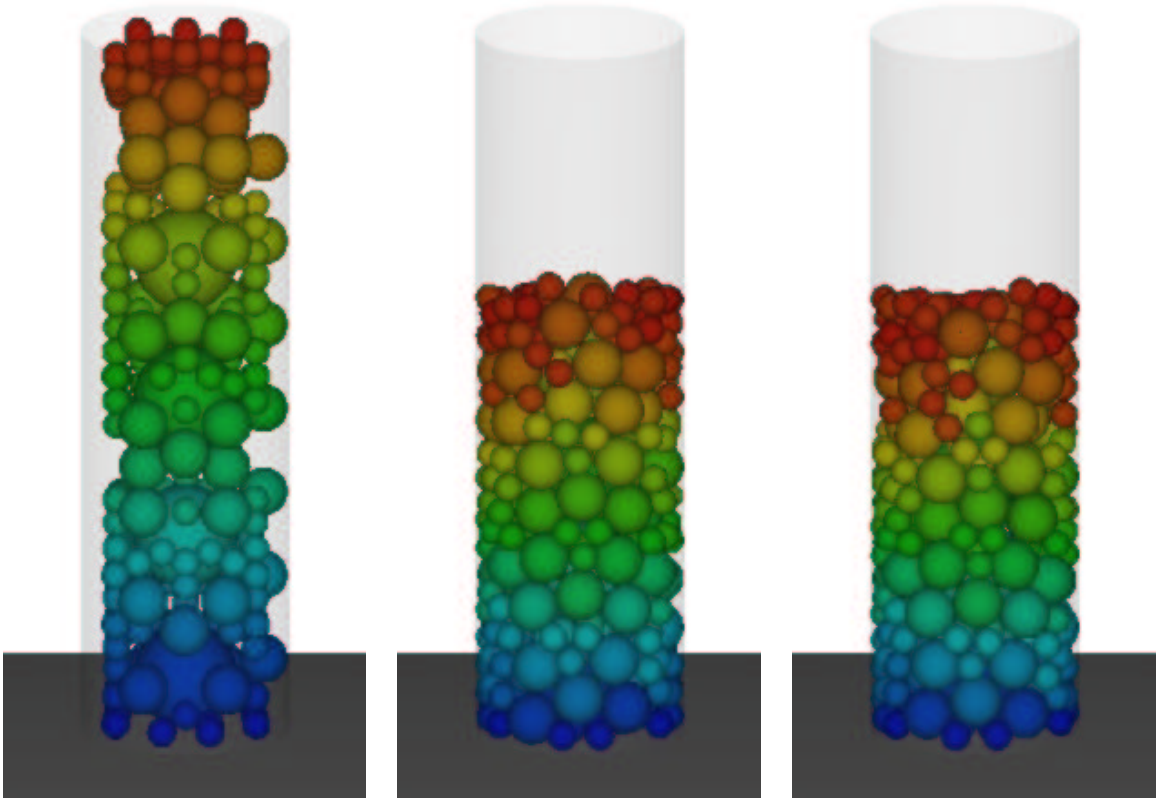


Figure 5.30: Simulation of the narrow distribution, initial distribution (left), after 1 second (center), after 10 seconds (right) in the 30mm cylinder.

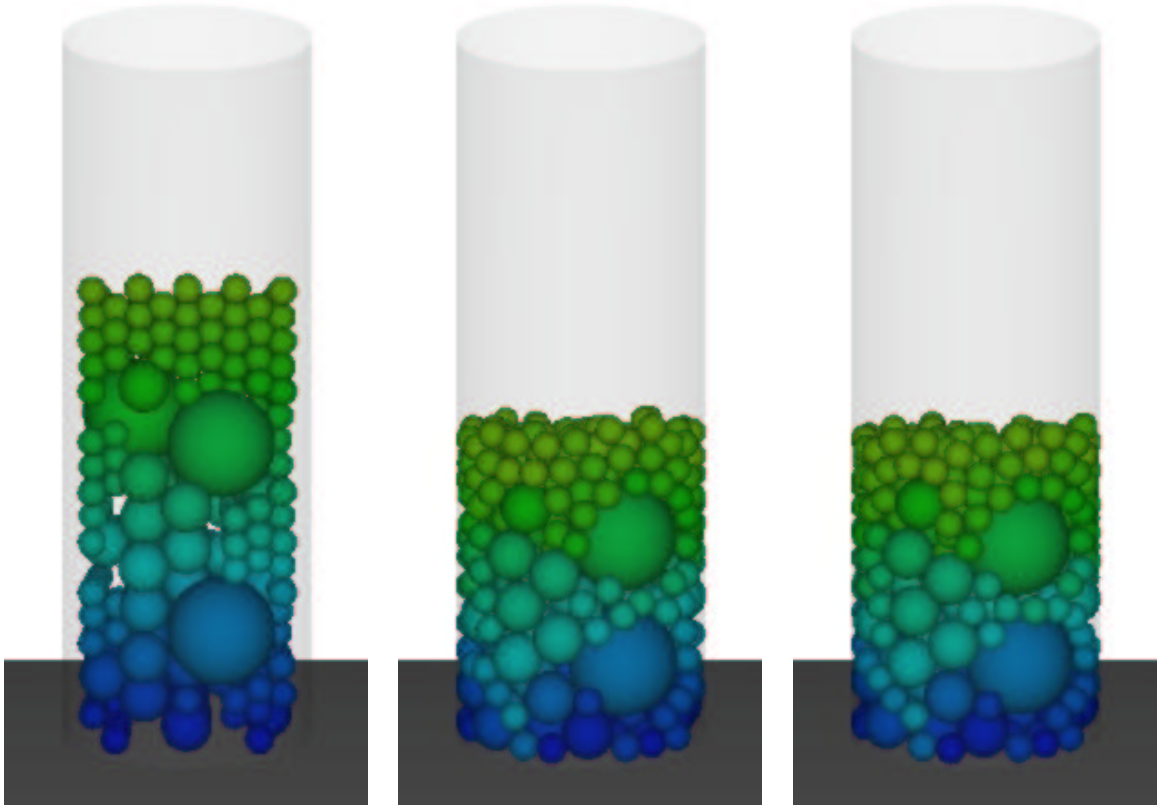


Figure 5.31: Simulation of the narrow distribution, initial distribution (left), after 1 second (center), after 10 seconds (right) in the 36mm cylinder.

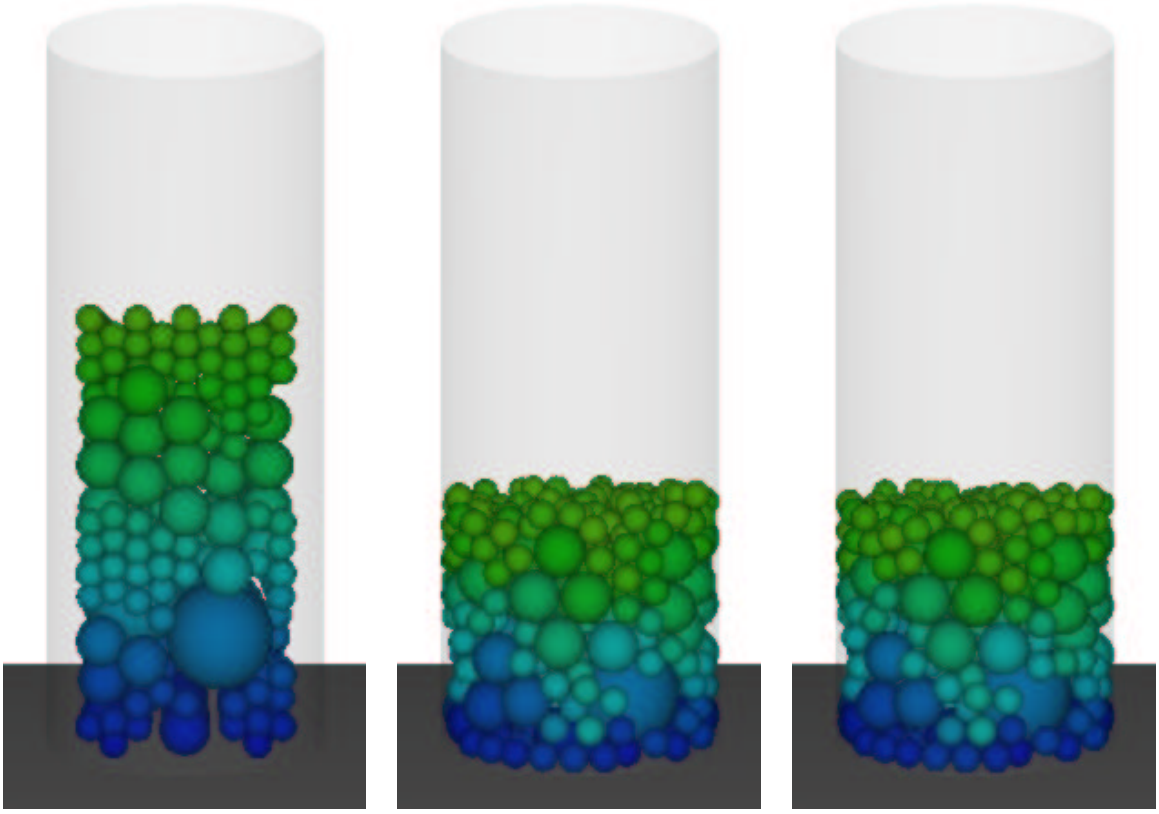


Figure 5.32: Simulation of the narrow distribution, initial distribution (left), after 1 second (center), after 10 seconds (right) in the 40mm cylinder.

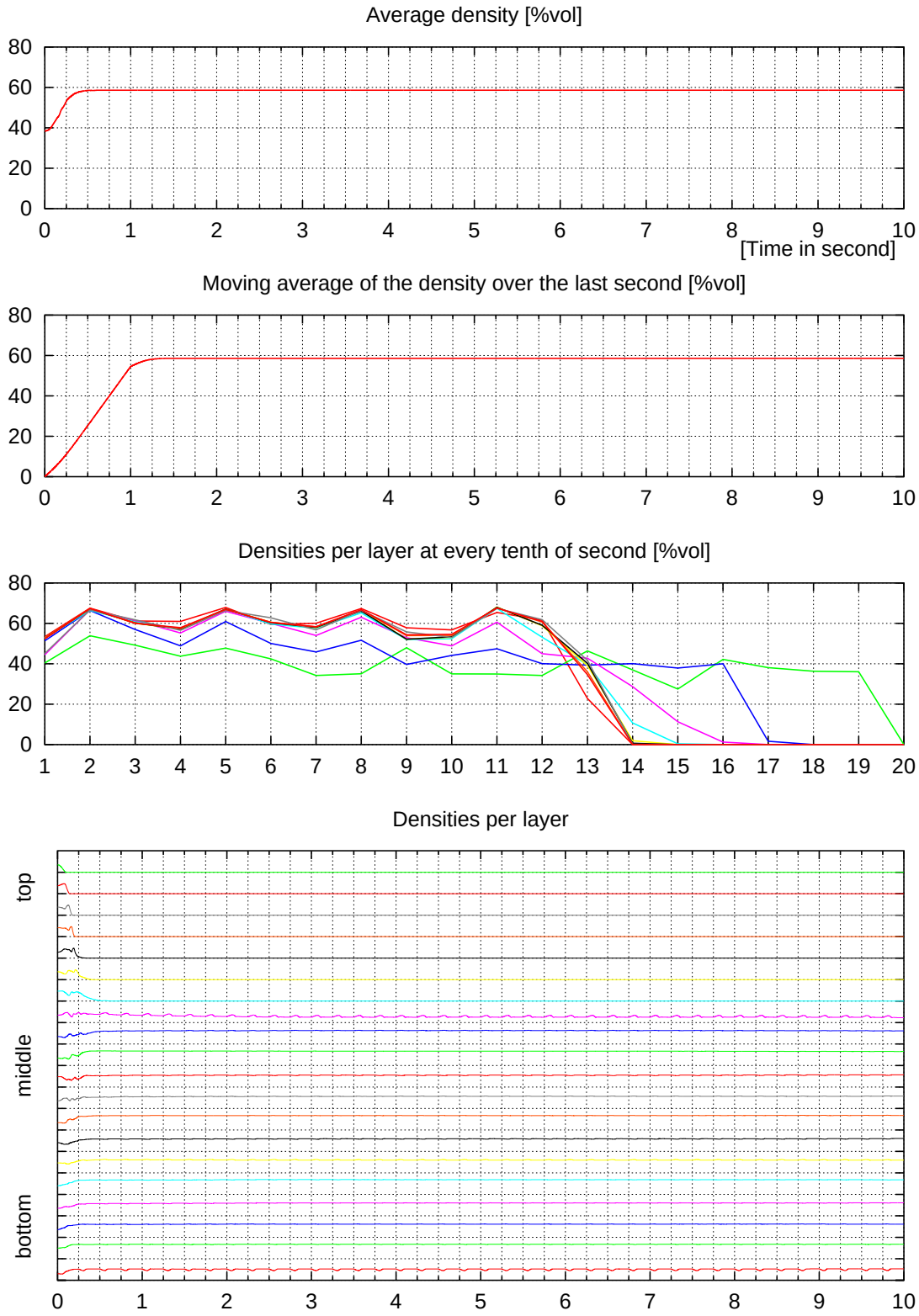


Figure 5.33: Densities for the narrow size distribution in the 30mm cylinder.

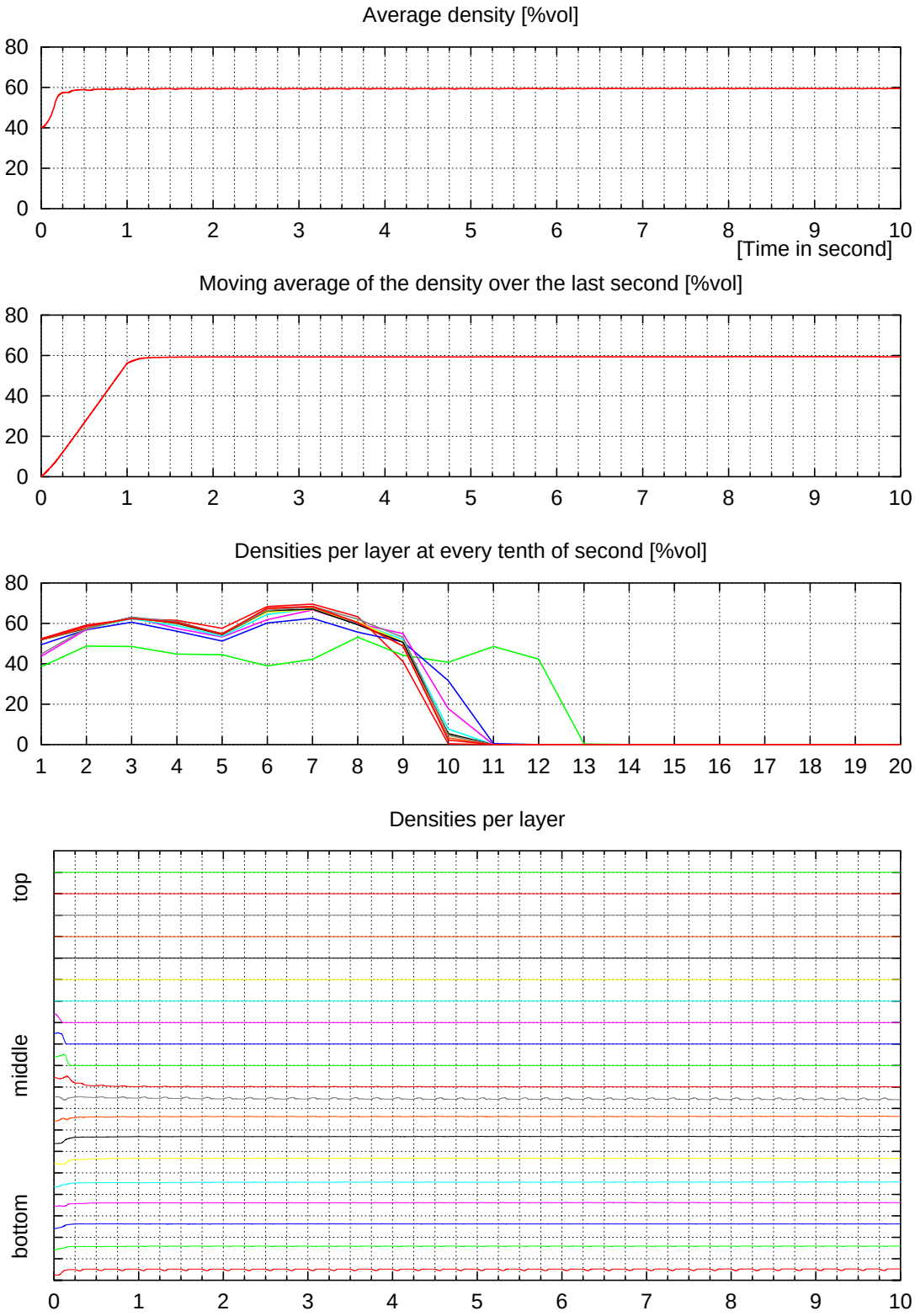


Figure 5.34: Densities for the narrow size distribution in the 36mm cylinder.

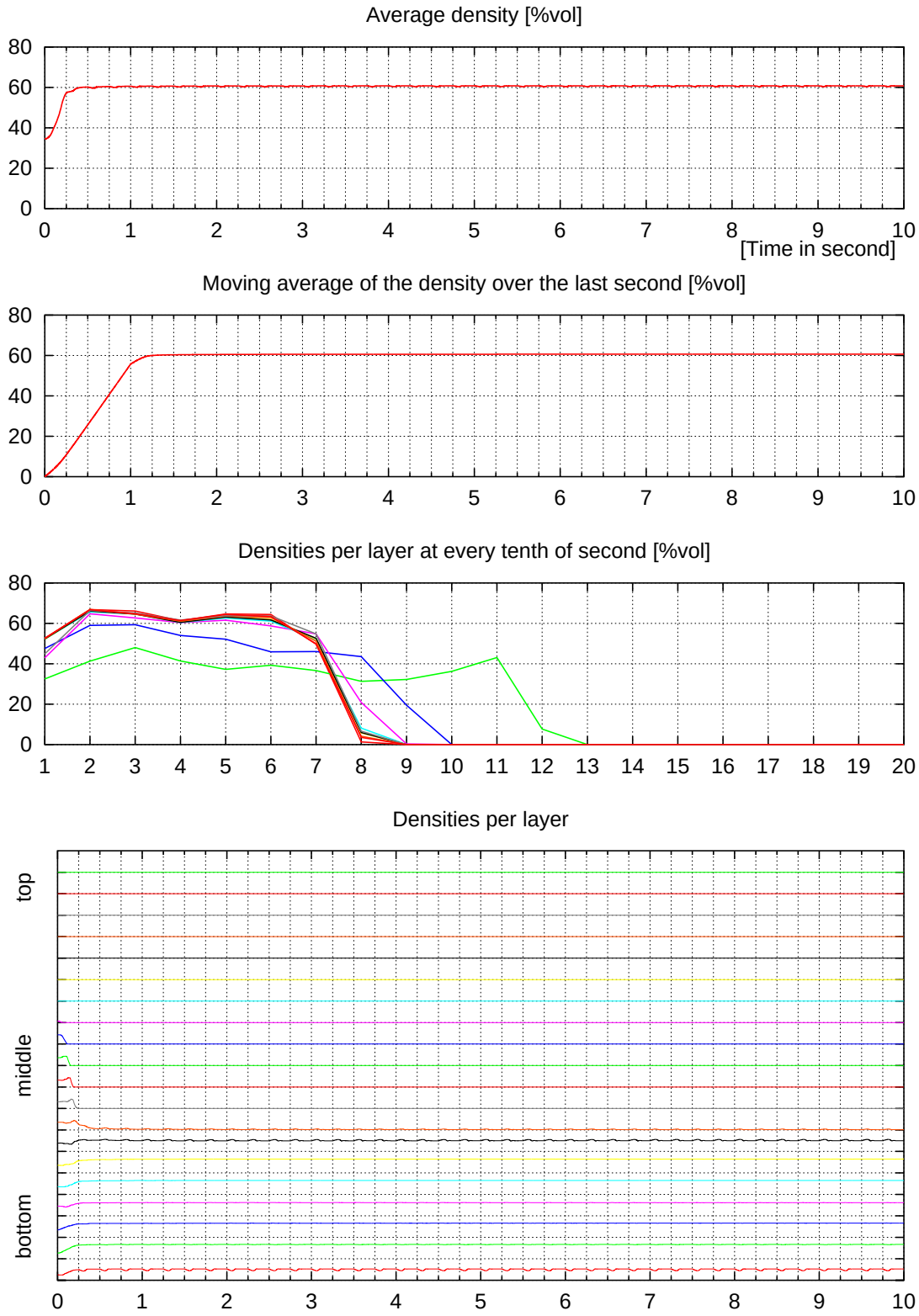


Figure 5.35: Densities for the narrow size distribution in the 40mm cylinder.

5.8.3 Wide distribution

The last set of experiments was performed with 2 large spheres of diameter 25mm, 270 medium spheres of diameter 4mm, and approximately 8900 spheres of diameter 1mm. Again, the two cylinders of 30mm and 36mm were used, see figure 5.36.

The packing with the small cylinder seems quite good and reached a density of 81%. With the large cylinder, on the other hand, there are not enough medium and small sized grains to cover the large ones and important segregation effects appear. We have dropped that case, but it confirms that much care must be taken when selecting grains sizes and quantities, and container size.

The computer simulation of this last case suffered from the difficulty to obtain a good initial distribution of the medium and small grains. As can be seen in figure 5.37, the medium grains would not fit in the lower part but instead most of them ended up stacked above the large grains, then the small grains filled the bottom part. The vibrations could not improve this highly biased initial distribution.

An attempt was made to improve this simulation by removing the small grains in order to see if the medium ones would spread over the whole height of the assembly. Several snapshots are shown in figure 5.38 and the usual density graphs are given in figure 5.39

The simulation in this case could not match the experimental results. This illustrates one key problem of our approach: the generation of an initial grain distribution as close as possible to what is done in reality. However, we believe that this difficulty will have less impact on future simulations where the grain size will be scaled down in order to use more grains and to suffer less from boundary effects.

Figure 5.36: Spheres of 1mm, 4mm and 25mm in cylinders of 30mm (top, left and right) and 36mm (bottom).

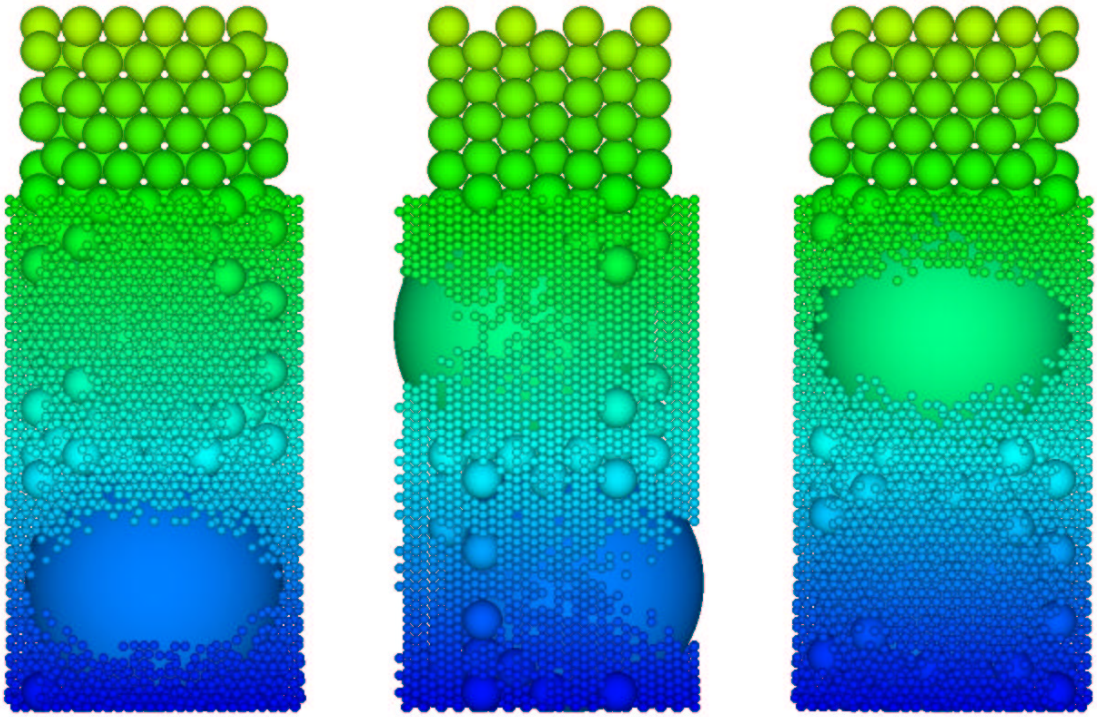


Figure 5.37: Simulation of the wide distribution: front, side and back view of the highly biased initial grain placement.

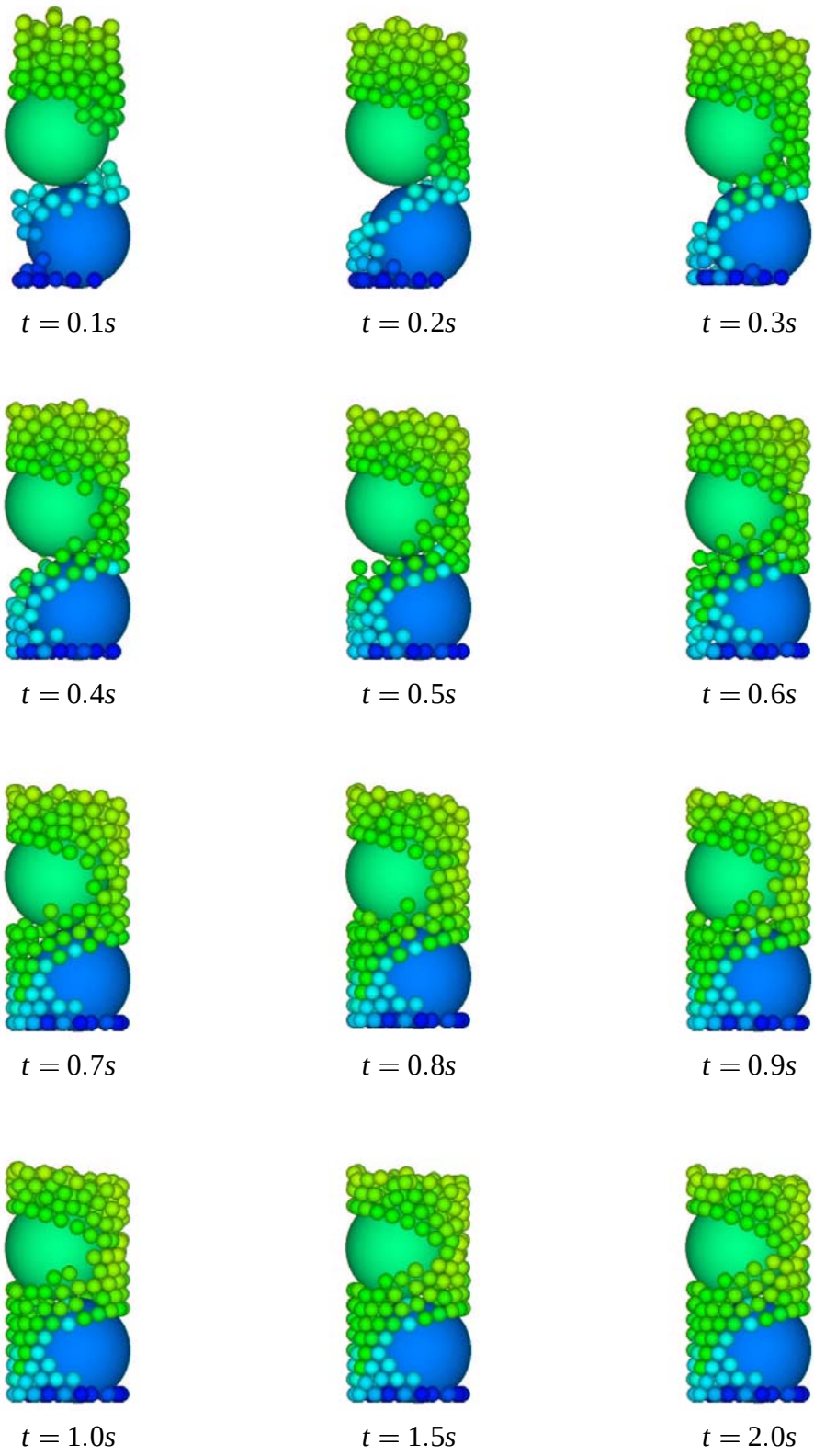


Figure 5.38: Side view of medium grains flowing around two large ones.

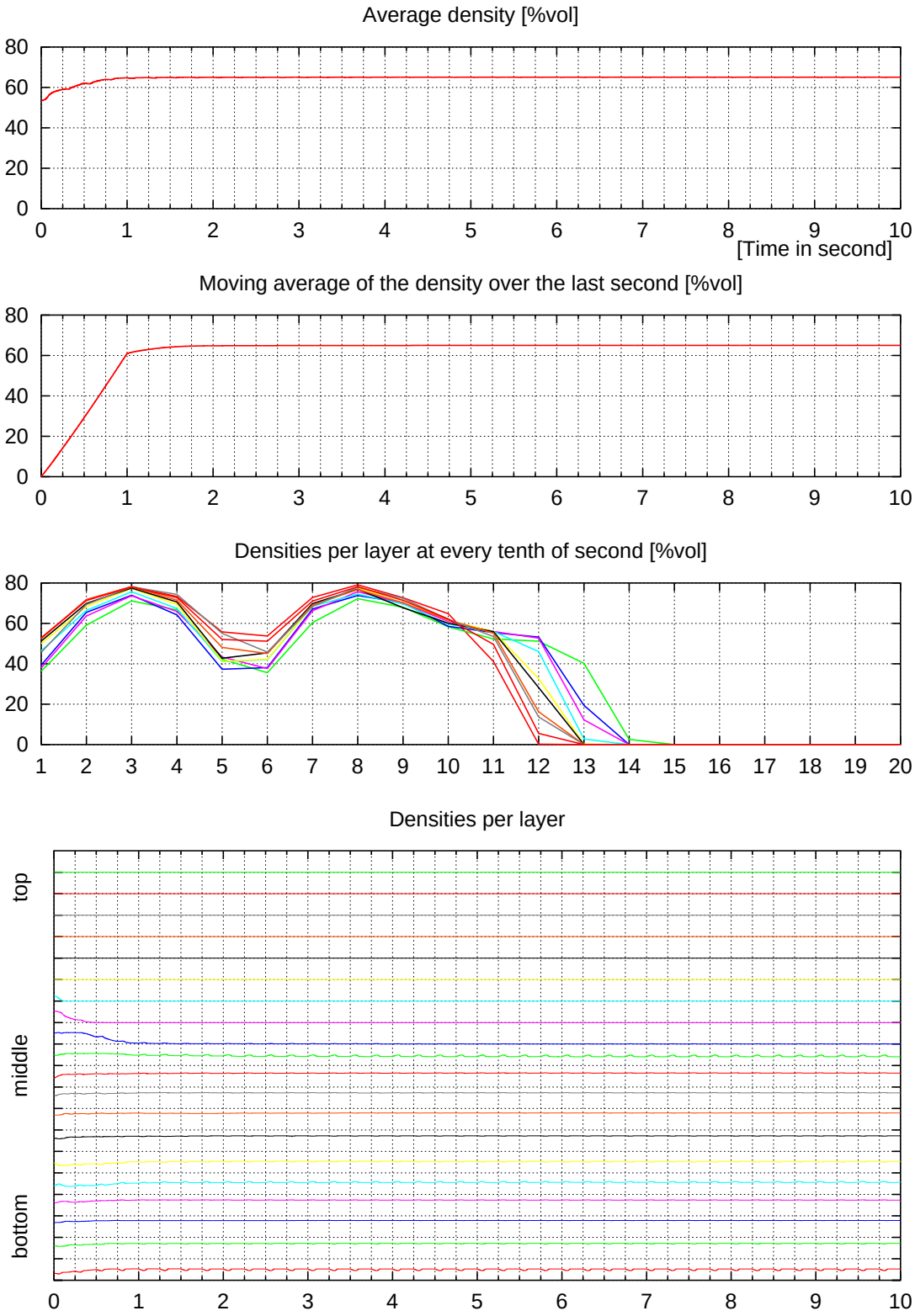


Figure 5.39: Densities for the large and medium grains.

5.9 Other possible approaches

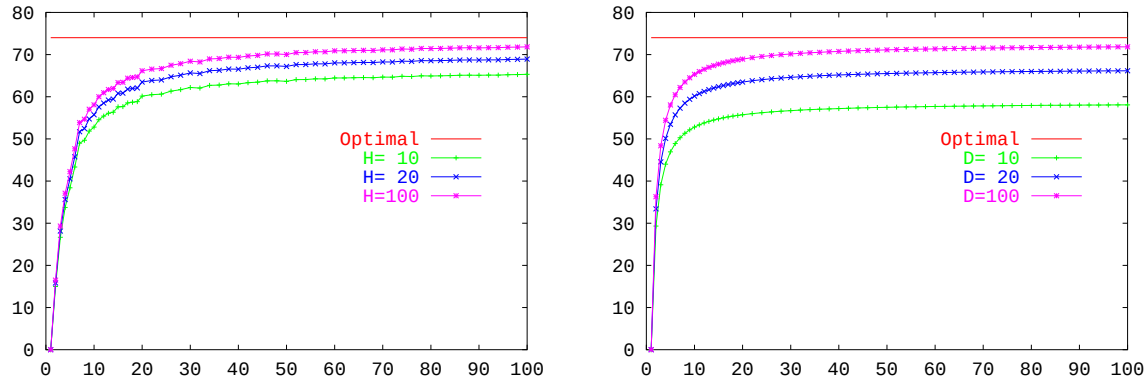
5.9.1 Face-centered cubic packings

Face-centered cubic packings attain the maximal density of all infinite packings of equal sized spheres. However, this result is of little help for our purpose. The gap between the face-centered cubic packing result and a dense packing of grains of various sizes is at least twofold: the container is bounded and the grains have different sizes.

5.9.1.1 Bounded vs. unbounded container

The face-centered cubic packing is optimal in an unbounded container. When boundaries are introduced, the situation changes. For example, let us assume we are packing grains of diameter 1 into a *spherical* container of diameter D. For the degenerate case where $D < 1$, we have a density of 0 since no grain will fit. At $D = 1$, we have a density of exactly 1! From then on, as long as $D < 2$, the density will decrease at a rate of $\frac{1}{D^3}$ until it reaches $\frac{1}{8}$, since only one grain will fit in a growing container. At $D = 2$ there is again a discontinuity since two grains will fit, yielding a density of $\frac{1}{4}$. And so on. The Kepler conjecture only says that for $D = \infty$, the density is equal to $\frac{\pi}{\sqrt{18}} \approx 0.74048$. Replacing a spherical container by anything else — cubic, cylindrical,... — only adds to the number of different configurations due to the loss of symmetry.

One would expect that the loss of density due to boundary effects decreases when the ratio between the grain size and the container size increases. This has been verified by observing the density reached by packing spheres of radius 1 in a cylinder of diameter D and height H, for various values of D and H. The results are shown in figure 5.40.



Growing D for fixed values of H — Growing H for fixed values of D

Figure 5.40: Densities obtained by growing the diameter (left) or height (right) of the cylindrical container.

This relation between the grain size – cylinder size ratio and the density has also been observed experimentally by Folly (2000) as shown in table 5.7.

grain diameter	30mm cylinder	36mm cylinder
25mm	47%	35%
15mm	46%	52%
7mm	60%	52%
4mm	58%	57%
1mm	66%	64%

Table 5.7: Experimental densities for various grain size / cylinder size ratios.

Let us now consider given, hopefully large, values of D and H, and the portion of an infinite face-centered cubic packing that fits in this cylindrical container[4]. By just shifting the center of the container by at most 1 (the diameter of the grains) in any dimension, the cutout made in the infinite packing will change, yielding different density values. This effect is shown in figure 5.41 where the density is plotted with respect to a horizontal shift in the XY plane perpendicular to the cylinder.

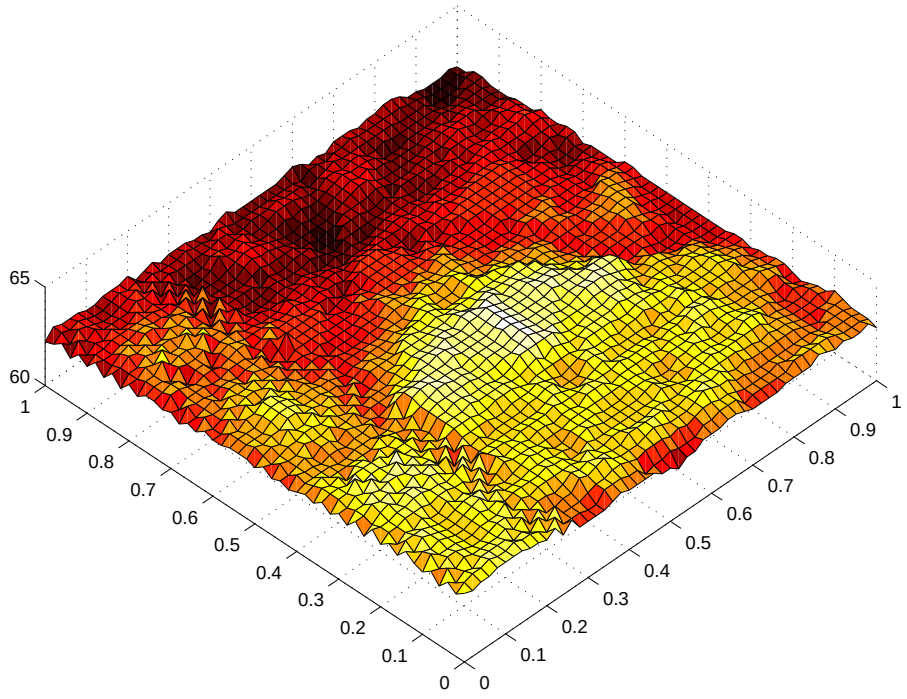


Figure 5.41: Density variations due to the variation of the alignment of the face-centered cubic packing and the cylinder.

One can easily convince oneself that this procedure is not guaranteed to give the highest densities, since the gaps near the boundaries can be quite large: Thus, even with grains of equal size, Kepler's packing may not be optimal in a bounded receptacle.

Interpolating these figures to the packing of powders is hazardous, but one can expect that all the results obtained in this current study for grains 1000 times larger than those of the real powders suffer more from boundary effects and therefore achieve slightly lower densities. Future simulations involving more smaller spheres will most certainly confirm this expectation.

[4] A picture of such a packing can be found on page 83.

5.9.1.2 Different grain sizes

When grains of two different sizes are considered, even in the case of an infinite container, one could be tempted to optimally pack the larger grains, then fill up the remaining gaps with the smaller ones. This however may not be optimal if the small grains do not fit exactly in the gaps. If we consider the optimal density as a function of the relative proportion of large and small grains, the Kepler conjecture says that at both ends of this relative proportion — that is only large or small grains — we have again a density of $\frac{\pi}{\sqrt{18}}$. Between these two extremes, virtually anything can happen, especially if a certain homogeneity is expected.

The optimization problem — select the proportion of large and small grains that gives the highest density — also remains open.

5.9.2 Exact computation

Another possible approach for computing the optimal density of a packing of spheres of various radii would be to consider the x-, y-, and z-coordinate of each sphere in a cylinder, write a set of constraints to enforce non-overlapping, and maximize the density expressed as a function of all the sphere coordinates.

Let N be the number of spheres. For each sphere i, its radius r_i is given, and the coordinates x_i, y_i, and z_i are the decision variables of the problem. Two spheres i and j are non-overlapping if

$$\sqrt{(x_i-x_j)^2+(y_i-y_j)^2+(z_i-z_j)^2} \quad \geq \quad r_i+r_j \tag{5.1}$$

Sphere i is in the vertical cylinder of radius R and of height H if

$$\sqrt{x_i^2+z_i^2} \quad \leq \quad R-r_i \tag{5.2}$$
$$y_i-r_i \quad \geq \quad 0 \tag{5.3}$$
$$y_i+r_i \quad \leq \quad H \tag{5.4}$$

The density can easily be computed as

$$d = \frac{4}{3R^2H}\sum_{i=1}^{N} r_i^3 \tag{5.5}$$

As expected, since R and r_i are fixed, increasing the density is obtained by minimizing H. We obtain the following problem:

$$\begin{array}{lll} \text{(P)} & \text{min.} & H \\ & \text{s.t.} & -(x_i-x_j)^2-(y_i-y_j)^2-(z_i-z_j)^2+(r_i+r_j)^2 \leq 0 \qquad \forall i,j=1,\ldots,N \\ & & x_i^2+z_i^2-(R-r_i)^2 \leq 0 \qquad \forall i=1,\ldots,N \\ & & -y_i+r_i \leq 0 \qquad \forall i=1,\ldots,N \\ & & y_i+r_i-H \leq 0 \qquad \forall i=1,\ldots,N \\ & & x_i,\, y_i,\, z_i,\, H \geq 0 \qquad \forall i=1,\ldots,N \end{array}$$

Problem (P) is a non-convex optimization problem with quadratic constraints and a linear objective function. Advanced resolution techniques combined with state-of-the art optimization software will produce an exact solution for problem (P) for very small cases with 10 to 20 spheres. Such a solution includes the individual position of each sphere in the assembly. This information is required for the application that inspired this formulation (see Dai and Sutou, 2000), but not in our case. Furthermore, problem (P) includes too many details by tracking each individual sphere instead of only considering 3 classes of grains. Therefore, we cannot consider this approach as a potential candidate for the study of optimal packings of powders.

5.10 Conclusion

Several simulations concerning the packing of irregular spheres were carried out and have served to fine-tune, calibrate and validate the DEM approach, the triangulation-based collision detection, the contact models and, last but not least, the simulation program. The results obtained cover several aspects such as the influence of the vibrations, the interaction between large, medium and small grains, and the boundary effects induced by the container. Whenever we were able to compare our simulations with experimental data we obtained similar behavior and numerical figures. In particular, the densities we measured were close to the values observed in practice.

Parameters introduced by the simulation method seemed not to influence the results with the notable exception of the initial position of the grains. The three-phase filling process we used, improved with a much finer control of the placement of the large grains, still introduces an undesirable bias that prevented us to replicate some experiments. A workaround could be to start the simulations by pouring grains in the cylinder instead of relying on them being already positioned.

The most exciting phrase to hear in science, the one that heralds new discoveries, is not 'Eureka!' (I've found it!), but 'That's funny...'.
– Isaac Asimov

Chapter 6

Other applications

6.1 Introduction

We present in this last chapter several other applications that have received less attention than the sphere packing problem. Pursuing the tradition of replicating by computer simulations real experiments performed at the Civil Engineering Department of EPFL, we teamed with B. Glisic of the Stress Analysis Laboratory to study the behavior of their deformation sensor for concrete structures (section 6.2). The traditional hourglass allowed us study some typical phenomena of granular flows (section 6.3). Finally, some preliminary validation simulations performed with clusters of grains are presented (section 6.4).

6.2 Sensor in concrete

This section was written in collaboration with B. Glisic of the Stress Analysis Laboratory, Civil Engineering Department, EPFL. His PhD thesis describes a deformation sensor for concrete at early and very early age. The problem at hand is to evaluate the perturbation on the measures induced by the sensor itself. These simulations were easily set up and confirmed the theoretical assumptions and experimental measures.

6.2.1 Concrete at early and very early age

Fresh concrete is a non-hardened multiphase mixture of different components. It consists of cement, water and aggregate (sand and gravel), but also contains a small amount of air. Furthermore, certain additives can be added to the basic mixture in order to improve or change certain characteristics of the concrete (workability, setting time etc.). The part consisting of cement and water is named cement paste. It is initially plastic, but gradually transforms into a solid

material. Complex physical and chemical processes commonly named the hydration process or just hydration cause this transformation.

Hydration is an exothermic process. Nevertheless, the heating of concrete does not have the same intensity during the whole hydration process. The most important heat release begins in the first few hours that follow the pouring of concrete. It reaches its hottest moments (depending of the type of cement) after 12 to 48 hours and cools afterwards (Legrand and Wirquin, 1994; Nonat and Mutin, 1994). The cooling of concrete is usually finished within 7 days, but can be longer depending on in-situ conditions and dimensions of the concrete elements. The period which begins with pouring and finishes when all thermal processes in concrete are calmed is considered here as the early age of concrete (Glisic, 2000a). The period, included in the early age, during which the concrete is still not hardened, is conventionally called the very early age (Glisic, 2000a).

Hydration has numerous consequences such as thermal process (heating and cooling), maturity, setting of the cement paste, hardening and solidification of the concrete as well as variations of mechanical properties such as Young modulus, Poisson's ratio, thermal expansion ratio and strength. An important consequence of the heating and cooling due to hydration is dimensional variation (deformation) of concrete elements.

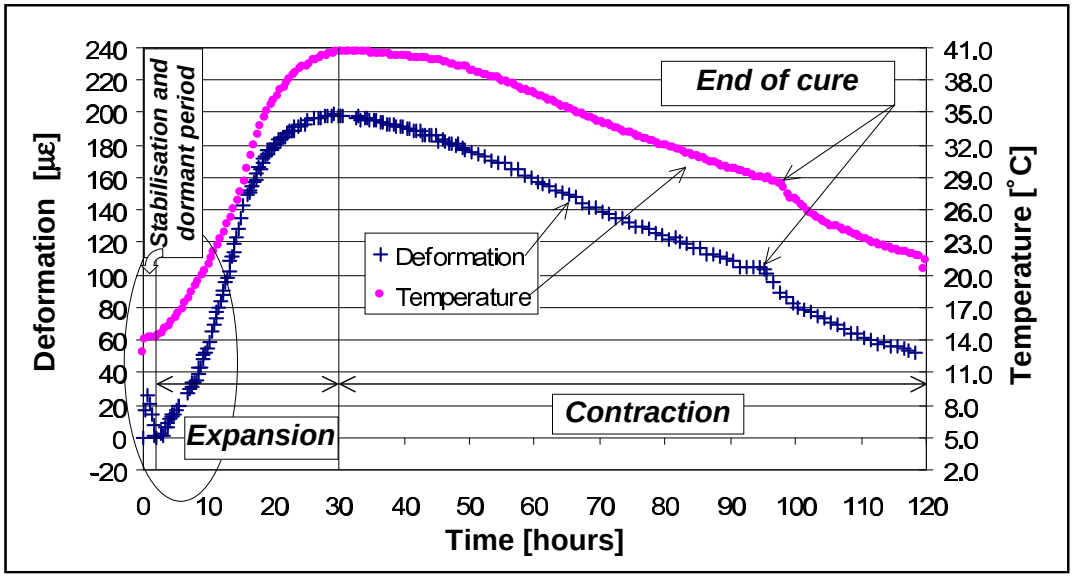


Figure 6.1: Typical ordinary concrete deformation and temperature evolution curves at early and very early age.

A typical diagram representing the evolution of temperature and deformation obtained by measurements is given in figure 6.1. The very early age deformation is distinguished in the encircled area.

Recent studies and research (Springenschmid, 1994; Tazawa and Iida, 1983) have shown that early age deformations can generate premature cracking of concrete that significantly increase the vulnerability of structures to noxious environmental influences. The cracks form "open doors" to the infiltration and penetration of noxious substances such as chlorides and sulphate water. These substances attack the concrete and rebars, and damage the structure, thereby reduc-

ing its long-term capacity and durability. That's why it is important to monitor the deformation of concrete at early and very early age.

6.2.2 The SOFO monitoring system

The deformation monitoring system named SOFO (French acronym of "Surveillance d'Ouvrage par Fibres Optiques" - "Monitoring of Structures by Optical Fibers") has been developed at the Stress Analysis Laboratory of the Swiss Federal Institute of Technology (IMAC-EPFL) (Inaudi, 1997). It is based on fiber optic technology and is capable of monitoring micrometer deformations over measurement bases up to a few meters. It is particularly well adapted to the measurement of structures built with conventional civil engineering materials (concrete, steel and timber). Since 1993 it has been successfully deployed in different types of structures such as bridges, tunnels and piles.

The standard SOFO (Glisic, 2000a) sensor is composed of two zones, the active zone which measures the deformations, and the passive zone that serves as the carrier of information between the active zone and the reading unit. The sensor is schematically represented in figure 6.2.

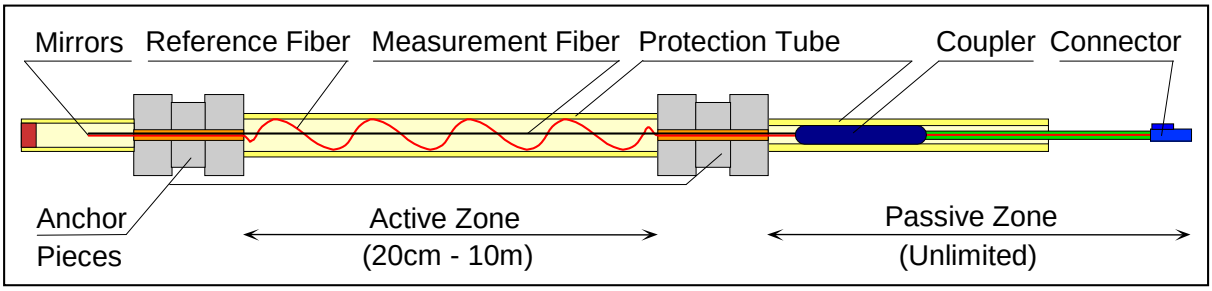


Figure 6.2: Standard SOFO Sensor

The active zone is limited by two anchor pieces and consists of two optical fibers placed in a protection tube. The anchor pieces have a double role: to attach the sensor to the monitored structure and to transmit the deformation from the structure to the sensing fibers. The measurement fiber is pretensioned between the anchor pieces in order to measure the shortening of the structure as well as its elongation. The reference fiber is independent of both the measurement fiber and the deformation of the structure, and its purpose is to compensate for temperature changes. The length of active zone of the Standard Sensor is comprised between 20 cm and 10 m. Out of these limits, it is difficult to guarantee the independence of the measurement and reference fibers.

The passive zone transmits the information from the active zone to the reading unit. It is composed of one single-mode fiber, a connector and a coupler, all protected by a plastic tube. The coupler is placed in the passive zone of the sensor, close to the anchor piece in order to increase the precision and to facilitate the manipulation during the measurement. The length of the passive zone can vary from several tens of centimeters to several tens of meters, depending on the need. If the passive zone is very long a simple fiber optic cable can extend it to up to 5 km. The protective plastic tube allows for easy manipulation, fast installation and very good protection of the sensors during the installation and long-term use of the system.

The SOFO sensors can be applied externally, attached to the surface of the structure, but also, and more importantly, they can be applied internally, embedded in fresh concrete. Installation of sensors before the pouring of concrete is a mandatory condition for deformation monitoring at very early age (before the hardening of concrete). The second condition is a good transfer of deformation from concrete to the sensor. SOFO sensors have a certain stiffness and it is difficult to estimate what is really measured during the very early age: the real deformation of concrete or something between the real deformation of concrete and the deformation of the sensor itself due to hydration heating. The simulations presented in this section address this aspect of the system.

6.2.3 Basic assumptions

In order to numerically evaluate the transfer of the deformation from the concrete at very early age, the following assumptions are accepted:

1. The initial stiffness of the very early age concrete is mainly generated by the aggregate skeleton (Glisic, 2000a). Therefore, the concrete is considered only as a granular medium, e.g. the influence of the cement paste is neglected.

2. The thermal deformation of the sensor's active zone itself is restrained by the aggregate skeleton, and therefore a force is generated in the protective plastic tube (see figure 6.2). To simplify the modeling, the problem is inverted: instead the heat load (caused by hydration) applied to the sensor and the aggregate simultaneously, only a force generated in the plastic tube is applied to the anchor piece, as shown in figure 6.3. The force magnitude of 3.304 N/°C is calculated using elasto-mechanic parameters of the protective plastic tube (Glisic, 2000a).

3. The transfer of deformation from the aggregate skeleton to the sensor is considered as good if the displacement of the anchor piece due to the applied force is less than 2 $\mu\varepsilon$/°C. This value is adopted since the thermal expansion coefficient of concrete at very early age is very elevated and varies between 40 $\mu\varepsilon$/°C and 10 $\mu\varepsilon$/°C. The resolution of the SOFO system is 2 μm and is independent from the length of the active zone. It means that the resolution of the measured strain is between 10 $\mu\varepsilon$ (for an active zone length of 0.2 m) and 0.2 $\mu\varepsilon$ (for an active zone length of 10 m).

Figure 6.3 shows a 2D schematic view of the numerical experiment.

The simulation code has been extended to support a cylindrical anchor piece inserted in the middle of the granular media. The anchor piece interacts with the surrounding grains, but its movements are restricted to translations (rotations are not permitted). Furthermore, the anchor piece applies a force of its own, thus inducing a reaction in the surrounding grains. That force comes from the dilatation of the active zone of the sensor due to the temperature raise during hydration. The anchor piece reports its position and forces acting upon it. From these values trajectories and force-displacement curves are drawn.

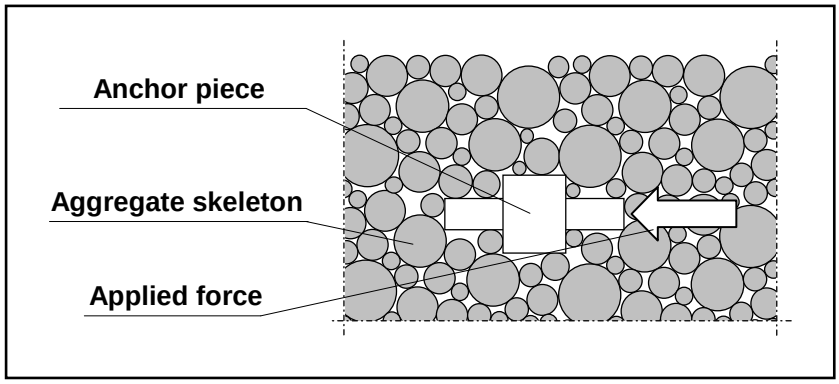


Figure 6.3: Schema of modeling

6.2.4 Experiments

The simulations are started by placing an anchor piece inside a cubic box and then filling the box with spherical grains of various sizes. In a first phase, the anchor piece is left at rest while the grains are packed under the influence of gravity. This phase lasts one second, which is enough for the grains to reach equilibrium. Then, an increasing force is applied by the anchor piece and its trajectory is observed.

The box is square, 200 mm by 200 mm, and filled by a layer of grains approx. 100 mm thick. The upper surface is free. The diameters of the grains range between 2 mm and 32 mm. The density of the grains is 2700 Kg/m^3.

The anchor piece is a cylinder 16 mm long and 20 mm in diameter, placed in the middle of the assembly. Its density is 7800 Kg/m^3. The force is exerted horizontally, along the main direction of the cylinder.

The force applied by the anchor piece increases with time according to the following rule:

$$F(t) = \begin{cases} \frac{\lceil (t-t_0)\text{nbstep} \rceil}{\text{nbstep}} F & \text{if } t \geq t_0 \\ 0 & \text{if } t < t_0 \end{cases}$$

where F is the reference force, reached after one second, t_0 is the time when the anchor piece starts pulling and nbstep is the number of constant steps to be made to reach this value. In the experiments, F is set to 2000 N and nbstep to 100, so that the force exerted by the anchor piece increased by 20 N every hundredth of second.

6.2.5 Numerical results and discussion

Various simulations were performed, with different grain size distributions that resulted in 2800 to 5500 individual grains. All have produced similar results, we discuss those of an average case, with 3909 grains.

Figure 6.4 shows the results with the sensor placed in the middle of the layer of grains, at a height of 50 mm. The trajectory of the anchor piece in the horizontal plane, along and perpendicular to its main direction is shown in the top graph. Remember that during the first second, the anchor piece is artificially fixed. Immediately after, there is a shift of about half a millimeter, and then the anchor piece rests in equilibrium for about 6 seconds. At that time the force is strong enough for the anchor piece to make its way further by shifting the surrounding grains.

This long equilibrium is particularly interesting. It means that even for a force up to 15'000 N, the anchor piece remains in place. Forces encountered in practice are at least one order of magnitude smaller. The initial displacement on the other hand is irrelevant for the real application.

The bottom graph of figure 6.4 shows the relation between the force and the displacement of the anchor piece. Low values of the force correspond to the initial shift, but if we look after that, from 200 N and on, we see that the slope of the curve is very low, that is the increase in the force induced by a raise of the temperature will have a minimal impact on the position of the anchor piece and thus will not introduce errors in the measures.

More precisely, a temperature raise of 30 °C generates a force increase of approximately 100 N, which in turn yields a displacement of less than 0.01 mm, as can be seen in figure 6.4 above 200 N. These 0.01 mm correspond to a value between 1.65 $\mu\varepsilon$ and 0.03 $\mu\varepsilon$ per °C depending on the length of the active zone, which has an impact of at most a few percent on the values measured by the sensor.

Figure 6.5 shows similar results when the sensor is placed near the bottom of the aggregate skeleton, at a height of 15 mm. We can observe again an initial shift, but the anchor piece not only settles quicker, but also remains even more stable than in the previous experiment, as shown by the almost flat curve of the bottom graph. Numerically, we obtain perturbations below the resolution of the sensor.

Figure 6.6 were obtained by placing the sensor near the top, again at 15 mm. Since the top is a free surface, it is much easier for the anchor piece to move in response to the force, as stated by the top graph of figure 6.6. However, the average slope between 0 N and 600 N in the bottom graph indicates that the perturbation would still be acceptable in this case.

6.2.6 Conclusion

Numerical modeling shows that the Standard Sensor is an appropriate device for the measurement of deformation of concrete at early and very early age. The difference between the measures and the actual deformation of the aggregate skeleton is estimated to be lower than the resolution of the monitoring system. This result is in accordance with results presented in (Glisic, 2000a). The absence of cement paste in the simulated experiments confirms that its mechanical influence can be neglected in the concrete at very early age, and opens new perspectives for the use of the SOFO sensors in granular materials.

Both the distinct element method and the simulation code for granular media based on spherical grains have again proved their versatility. Only minor modifications were required to perform efficient computer simulations of the SOFO system. Such simulations had not been possible with existing tools based on other approaches.

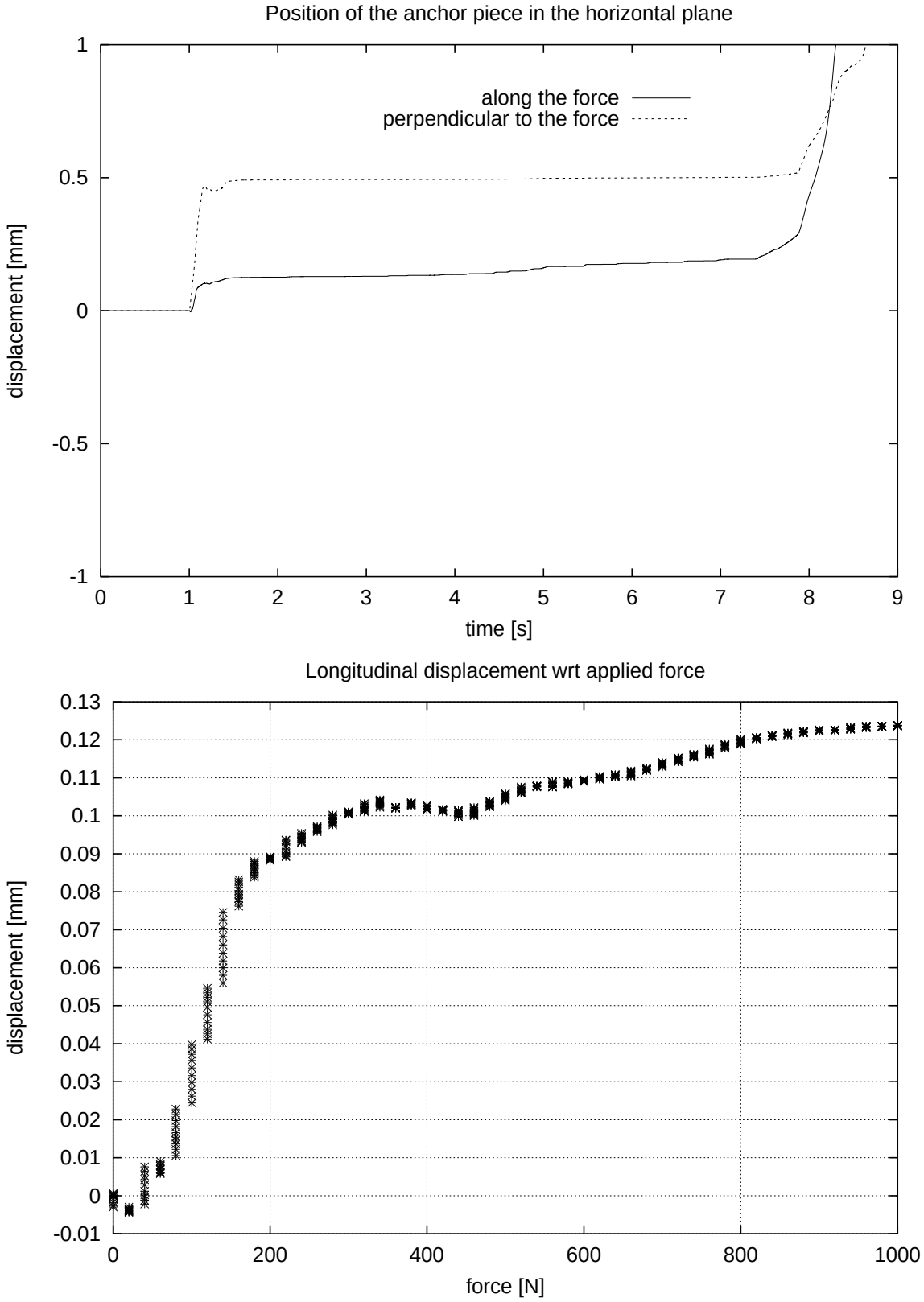


Figure 6.4: The sensor in the middle of the concrete.

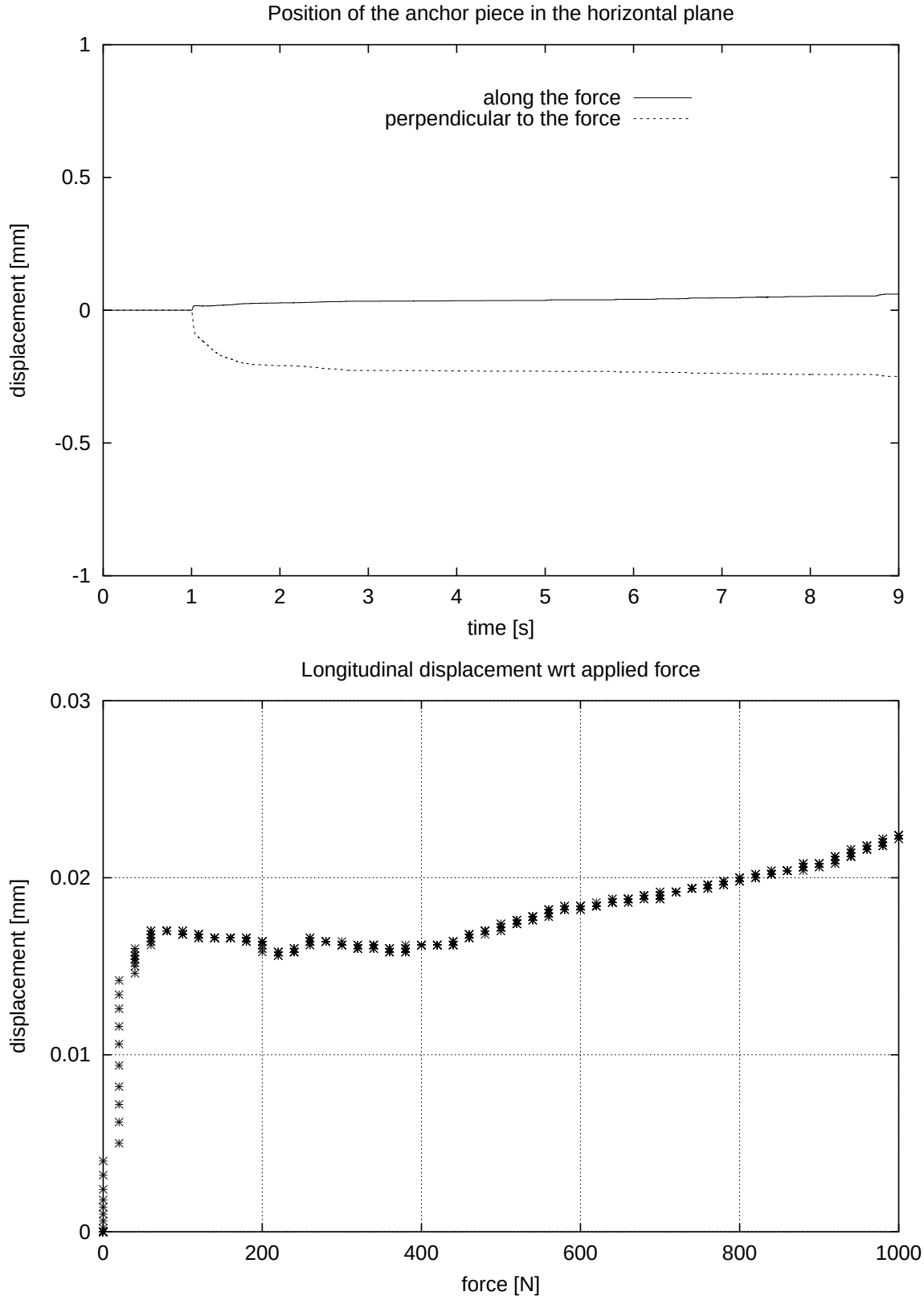


Figure 6.5: The sensor near the bottom of the concrete.

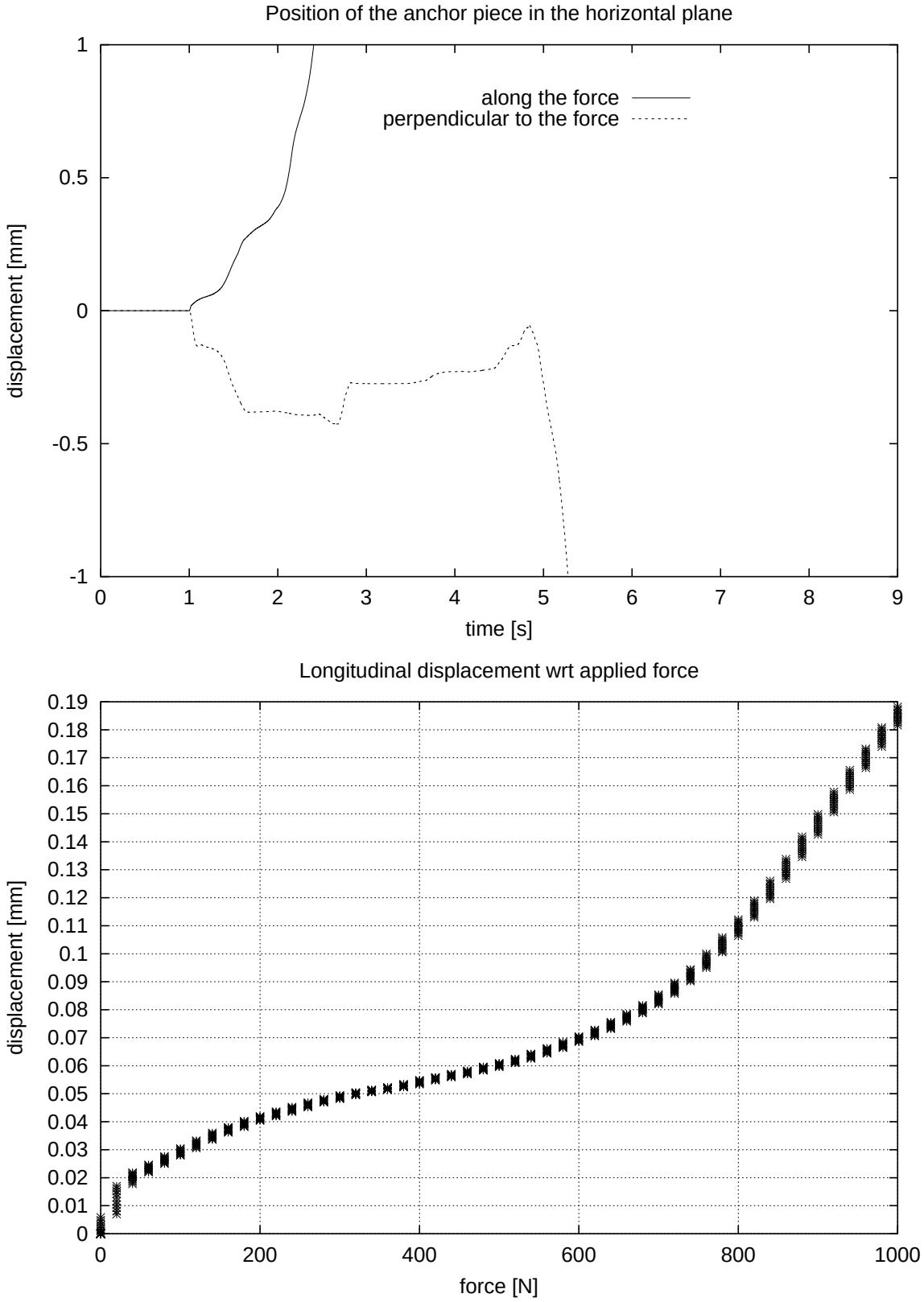


Figure 6.6: The sensor near the top of the concrete.

6.3 Hourglass flow

A classical example when dealing with powders and grains is the flow of granular material in an hourglass, as shown in figure 6.7. Even if this application does not map directly to an industrial problem, it exhibits several key phenomena:

- The slow movements and rearrangements of the grains in the upper part as they get closer to the slot
- The flow of grains though the slot
- The free flow of grains falling down in the lower part
- The impact of that flow with the ground
- The accumulation of the grains in the lower part, in particular their rearrangement into a pile.

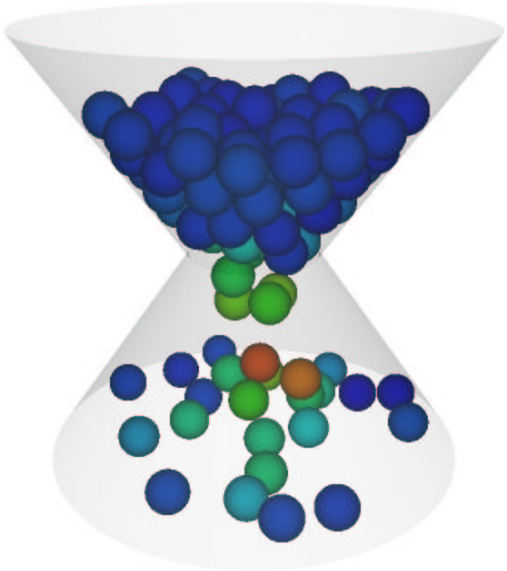


Figure 6.7: The flow of grains in an hourglass. The grains are colored according to their instantaneous velocity.

Individually, every one of these phenomena plays an important role in industrial processes dealing with granular flows. We performed several simulations with different grain sizes, different hourglass shape, and grains of two different sizes. By rotating the gravity, we could have a first approximation of what happens when the hourglass is rotated.

The dimensions of the hourglass are shown in figure 6.8. The simulation is prepared by filling the upper part with grains, and then the action of gravity will result in the falling of grains through the opening.

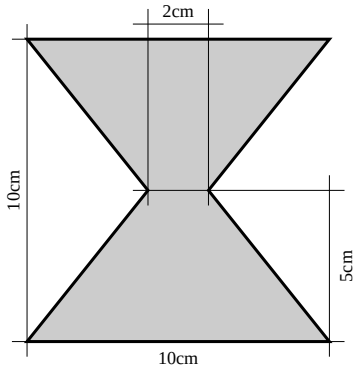


Figure 6.8: The shape and dimension of the hourglass.

6.3.1 Simple flow in a regular hourglass

With grains of equal size, one usually wants that the quantity of material flowing through the slot depends neither on time nor on the mass in the upper part of the hourglass. This important property is crucial for using the hourglass to measure time as used to be the case.

Three simulations where performed, they are summarized in table 6.1. Figures 6.9, 6.10 and 6.11 show the actual flow though the slot and a snapshot at three different times. In experiment A (figure 6.9), the flow is regular throughout the whole period. In experiment B (figure 6.10), there are too many grains — or more precisely the arrangement of the grains is not optimal in the lower part — and the slot is blocked. To circumvent this situation, experiment C (figure 6.11) was performed with less grains and the flow stopped earlier.

Name	number of grains	grain size [mm]
A	2790	4
B	7236	3
C	5898	3

Table 6.1: Summary of the three experiments.

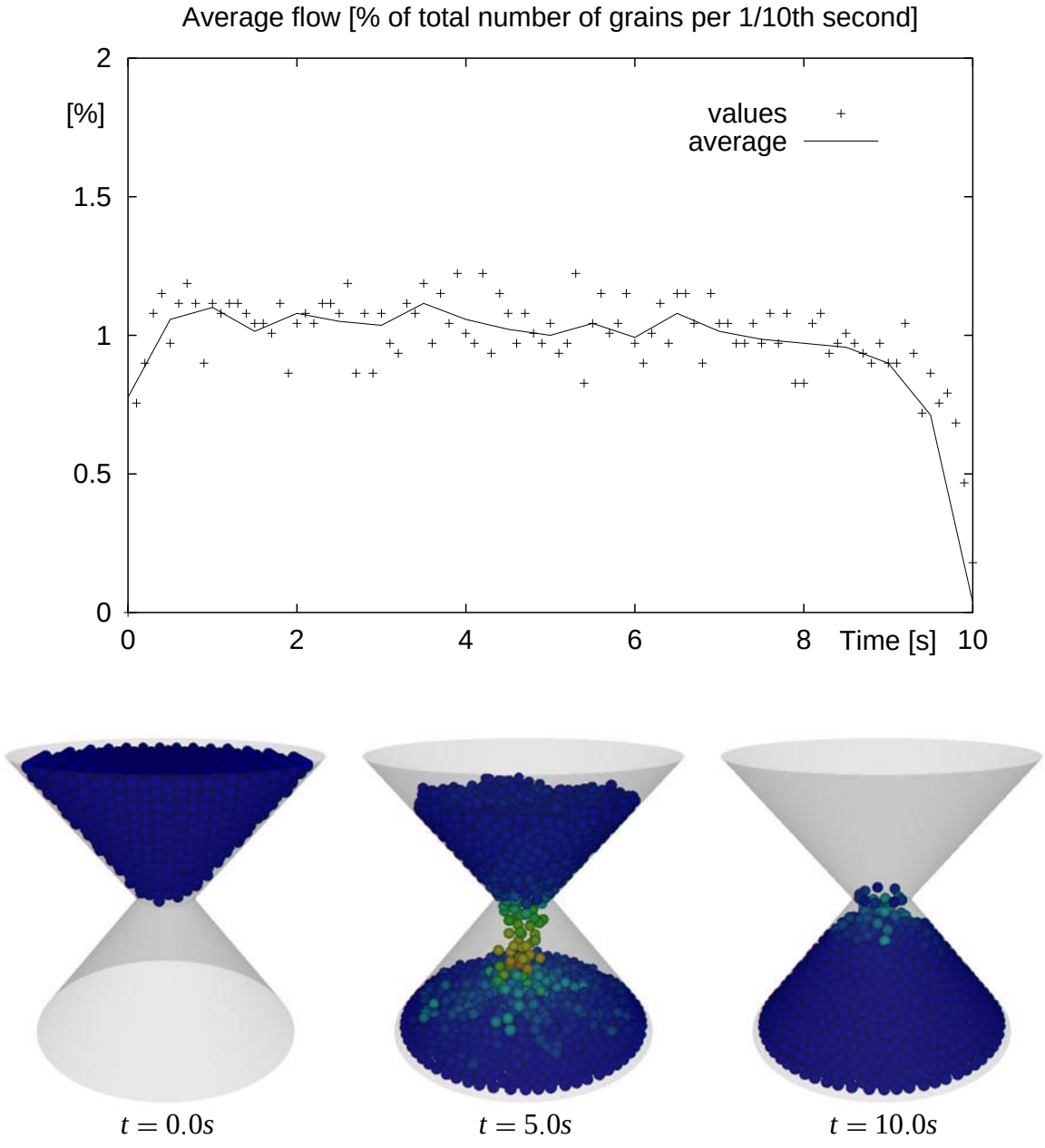
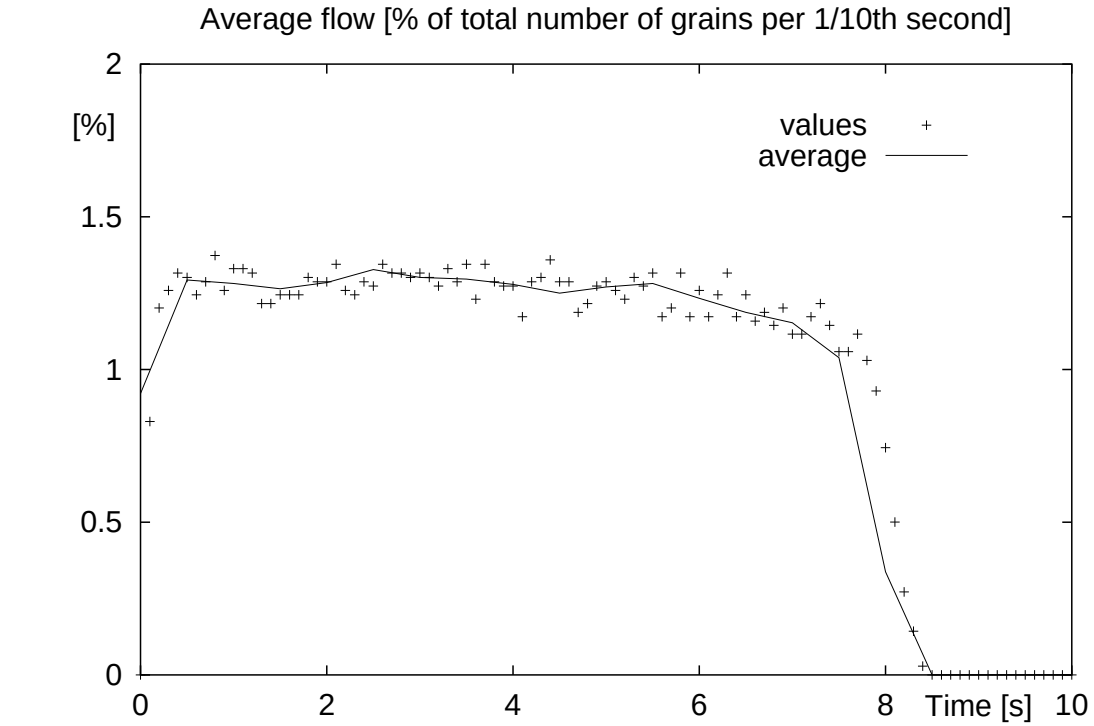
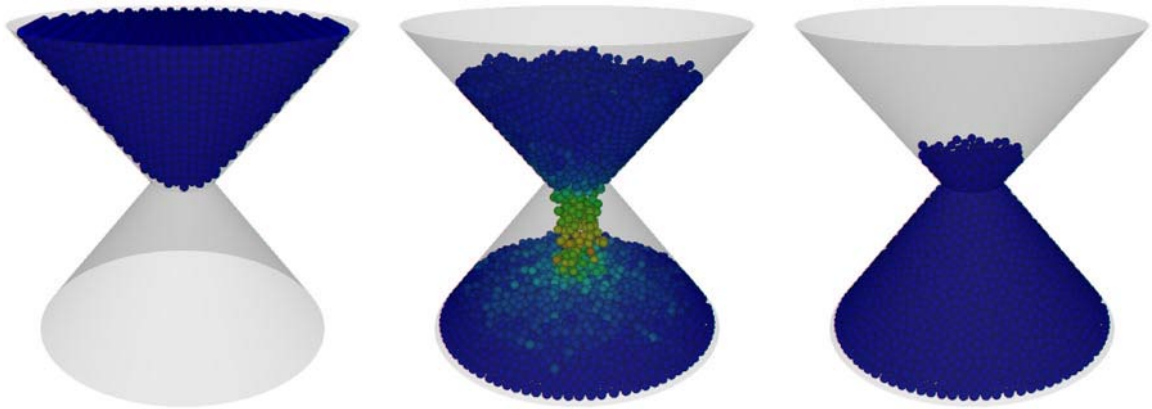


Figure 6.9: Experiment A.





$t = 0.0s$ $t = 5.0s$ $t = 10.0s$

Figure 6.10: Experiment B.

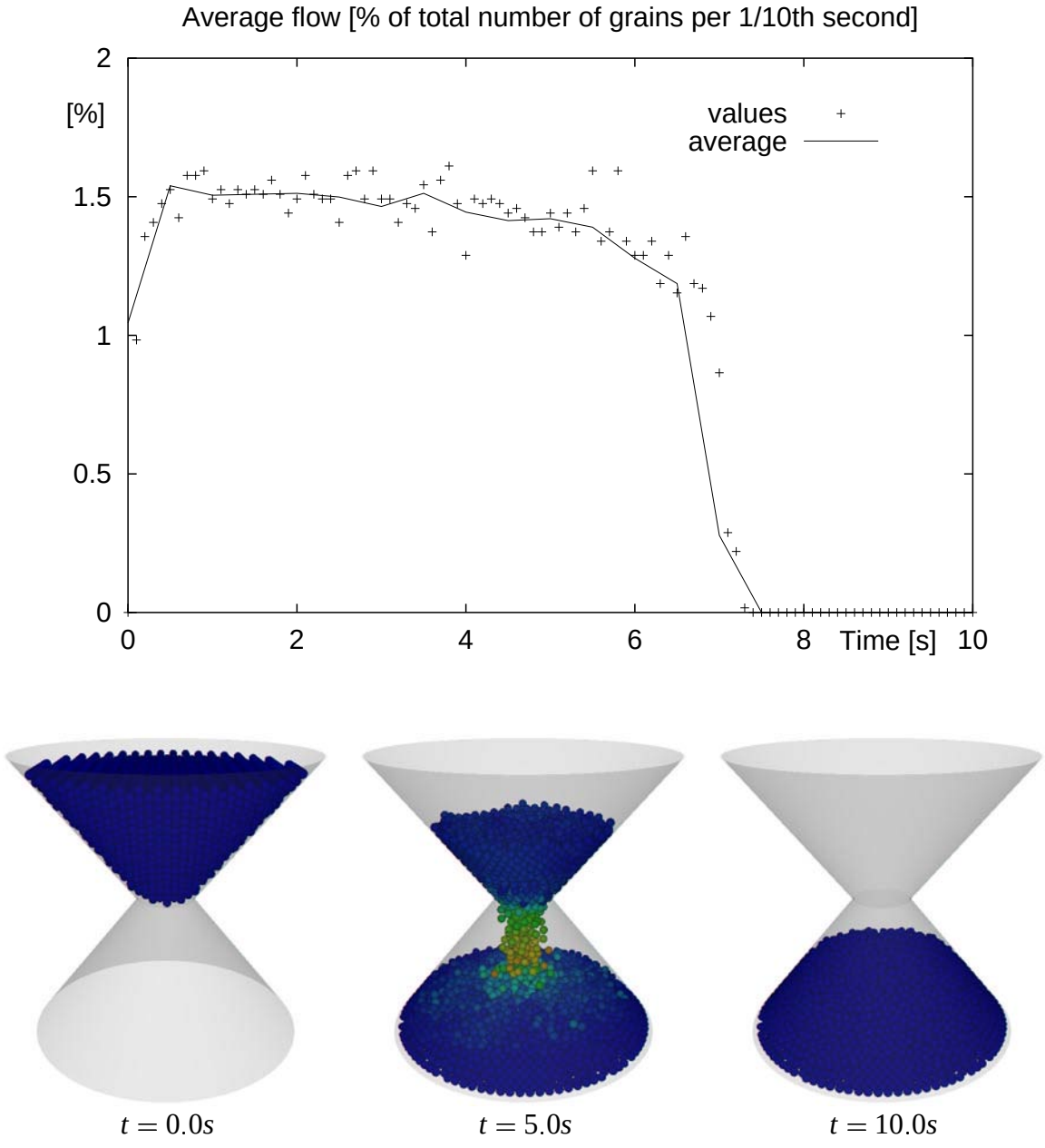


Figure 6.11: Experiment C.

6.3.2 Grains of different sizes

The same simulation was then run with grains of two different sizes, see table 6.2. This results in a much more chaotic flow, as can be seen in figure 6.12.

Grain type	number of grains	grain size [mm]	portion of mass [%]
large	273	6	61.67
small	4582	2	38.33
total	4855		

Table 6.2: Characteristics of the two grain types.

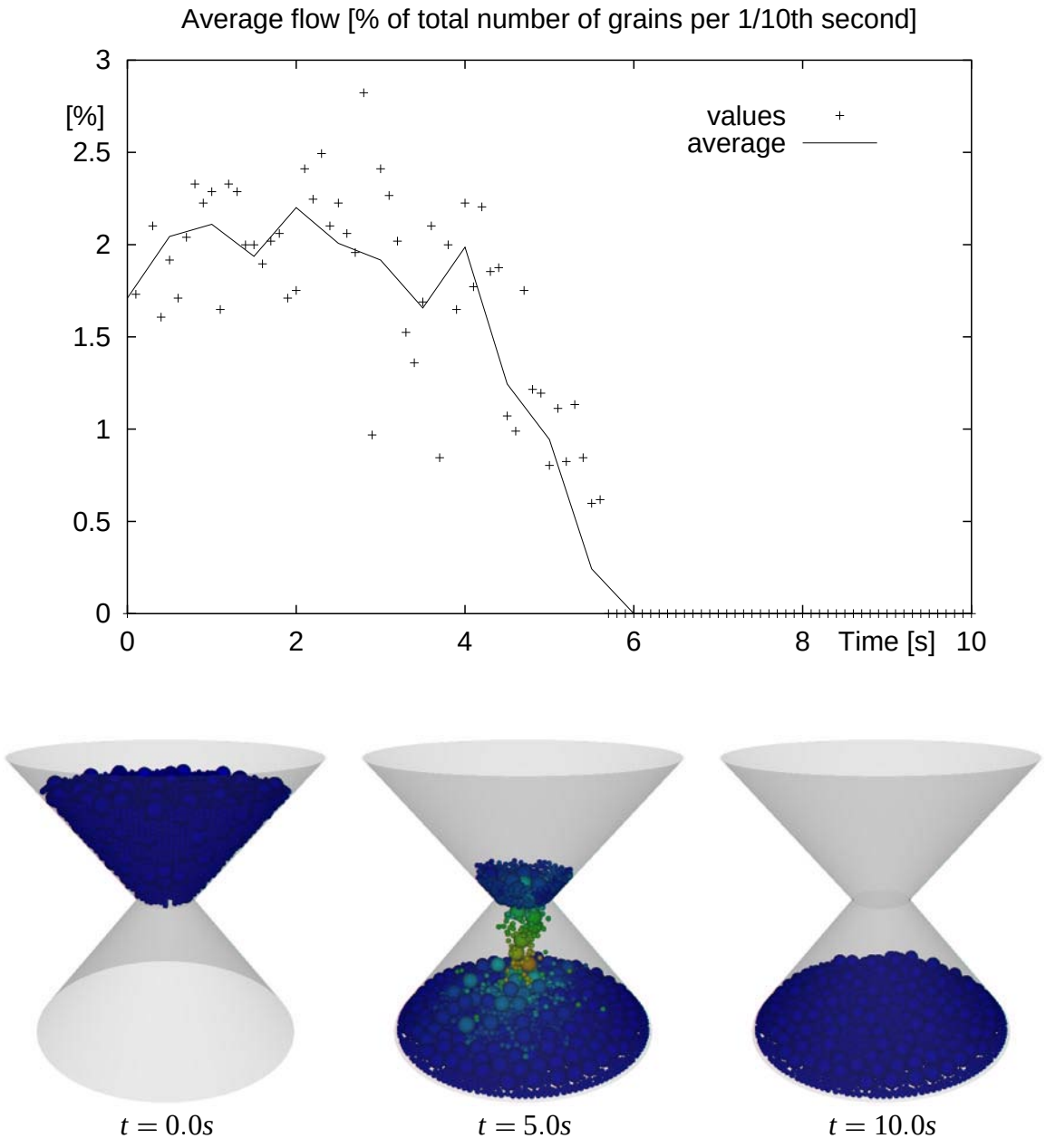
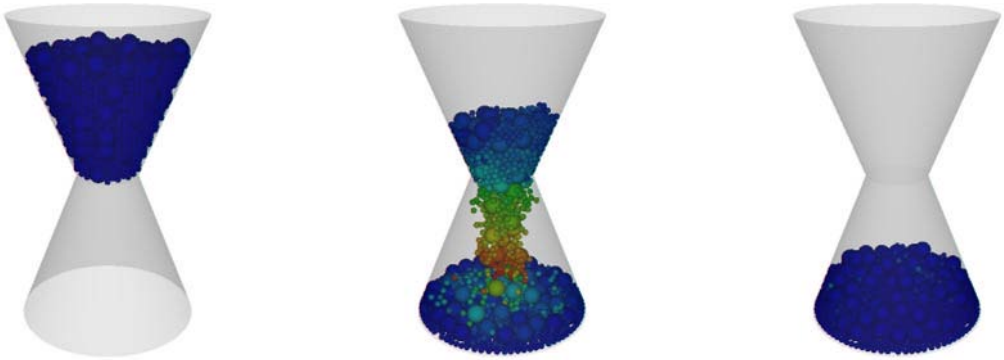


Figure 6.12: The flow with two grain sizes.

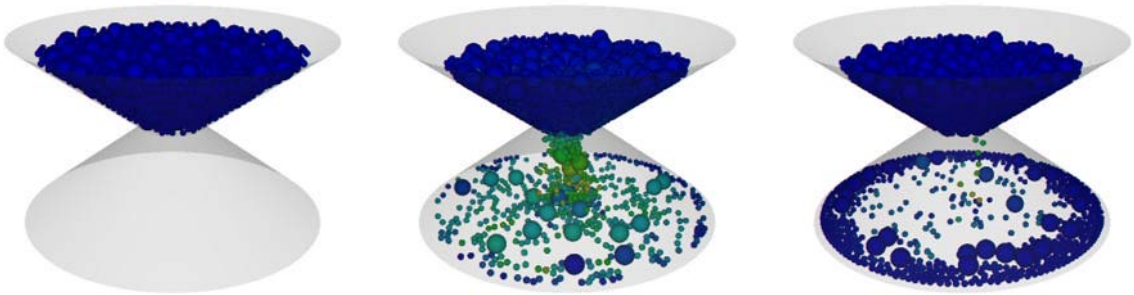
6.3.3 Various hourglass shapes

Similar sets of grains were used in different hourglasses, as shown in figure 6.13 and 6.14. The size distribution of the grains is the same, as is the diameter of the slot. In the first case, the width of the base is shrunk from 10cm to 6cm. The flow in this steep hourglass is very fast. In the second case the total height is shrunk from 10cm to 6cm. The flow is much slower and even stops when some large grains block the slot of the hourglass.



$t = 0.0s$ $t = 1.25s$ $t = 2.5s$

Figure 6.13: The flow with two grain sizes in a tall hourglass.



$t = 0.0s$ $t = 0.75s$ $t = 1.5s$

Figure 6.14: The flow with two grain sizes in a flat hourglass.

6.3.4 Rotating the hourglass

Two very small simulations were performed with 71 grains in a tall rotating hourglass. Twenty seconds were simulated, with rotations taking place at 4, 8, 12 and 16 seconds. In the first case (see figure 6.16), the rotation takes 0.1 second. When it is completed, the grains have barely started to move and thus fall down quite heavily. In the second case (see figure 6.17), the rotation takes one second and the grains flow smoothly along the inner wall, which induces a floating movement near the slot.

During one fast reversal, about a quarter of the grains remained blocked in the upper part of the hourglass, as shown in figure 6.15.

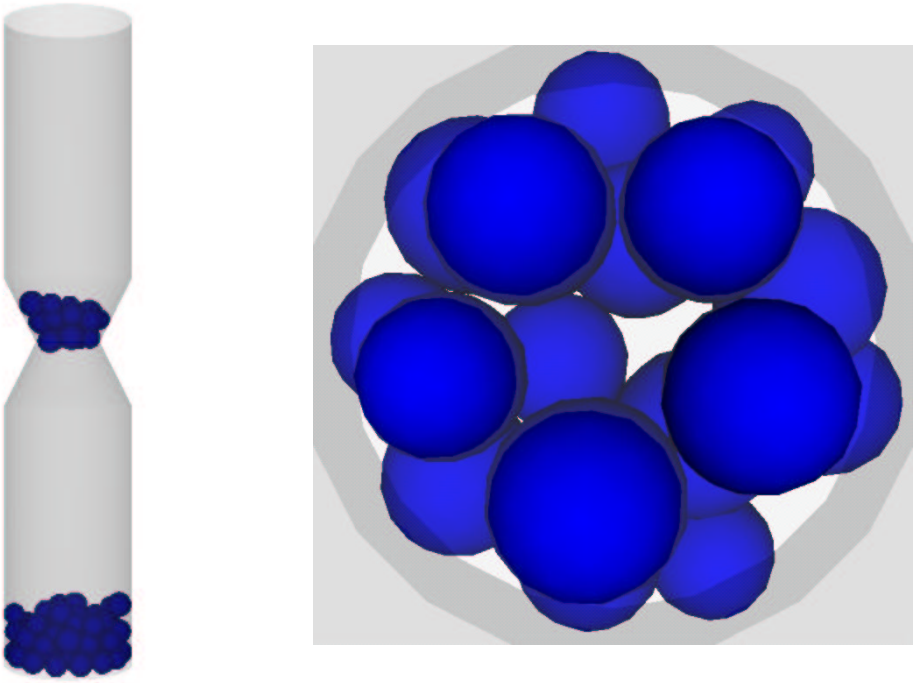


Figure 6.15: Grains blocked in the hourglass. *Left*: side view. *Right*: view from below the slot.

The tall hourglass of §6.3.3 was also rotated, as shown in figure 6.18. The rotation lasted half a second and snapshots were taken at 20 frames per second.

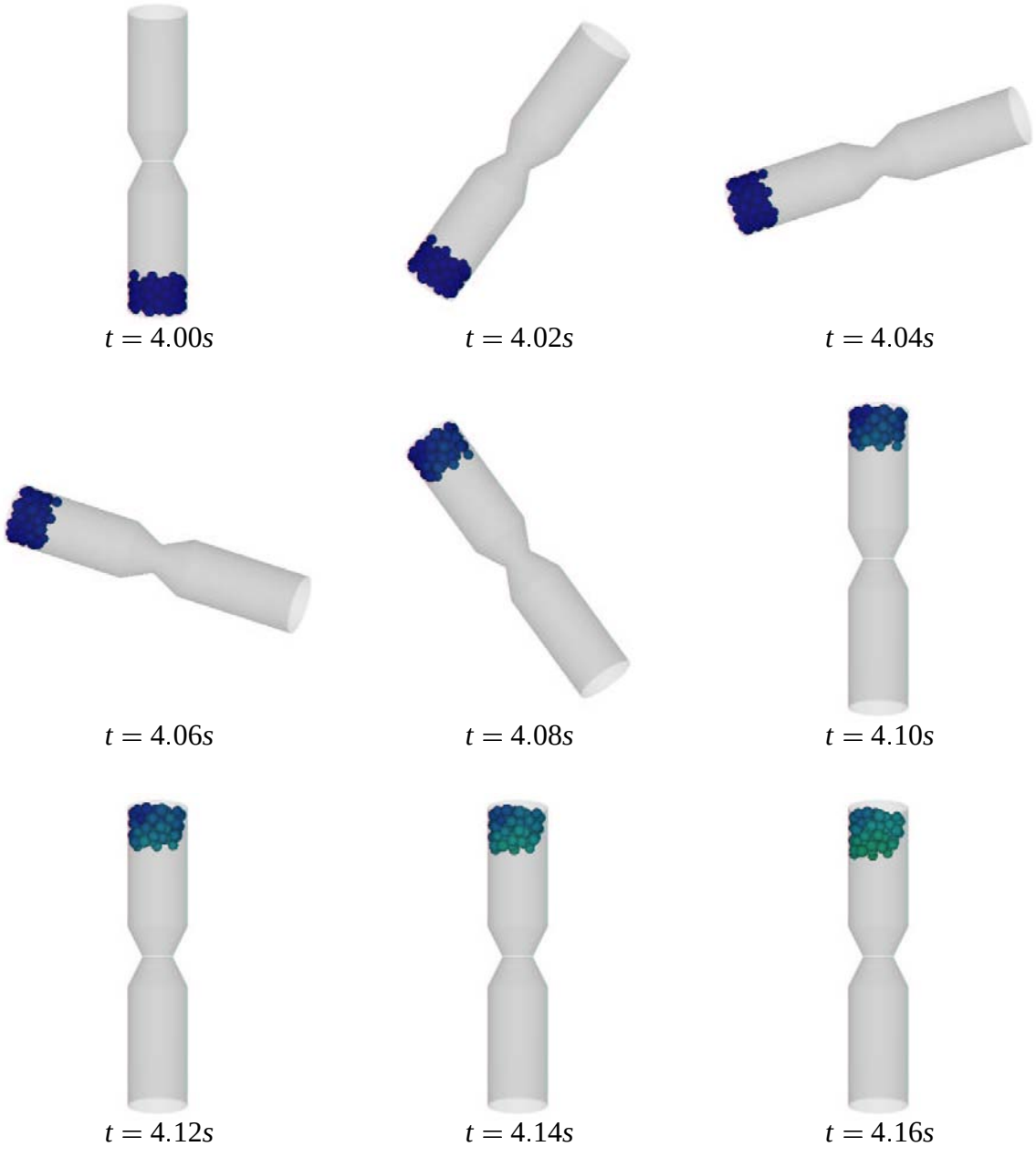


Figure 6.16: Individual snapshots during the fast reversal.

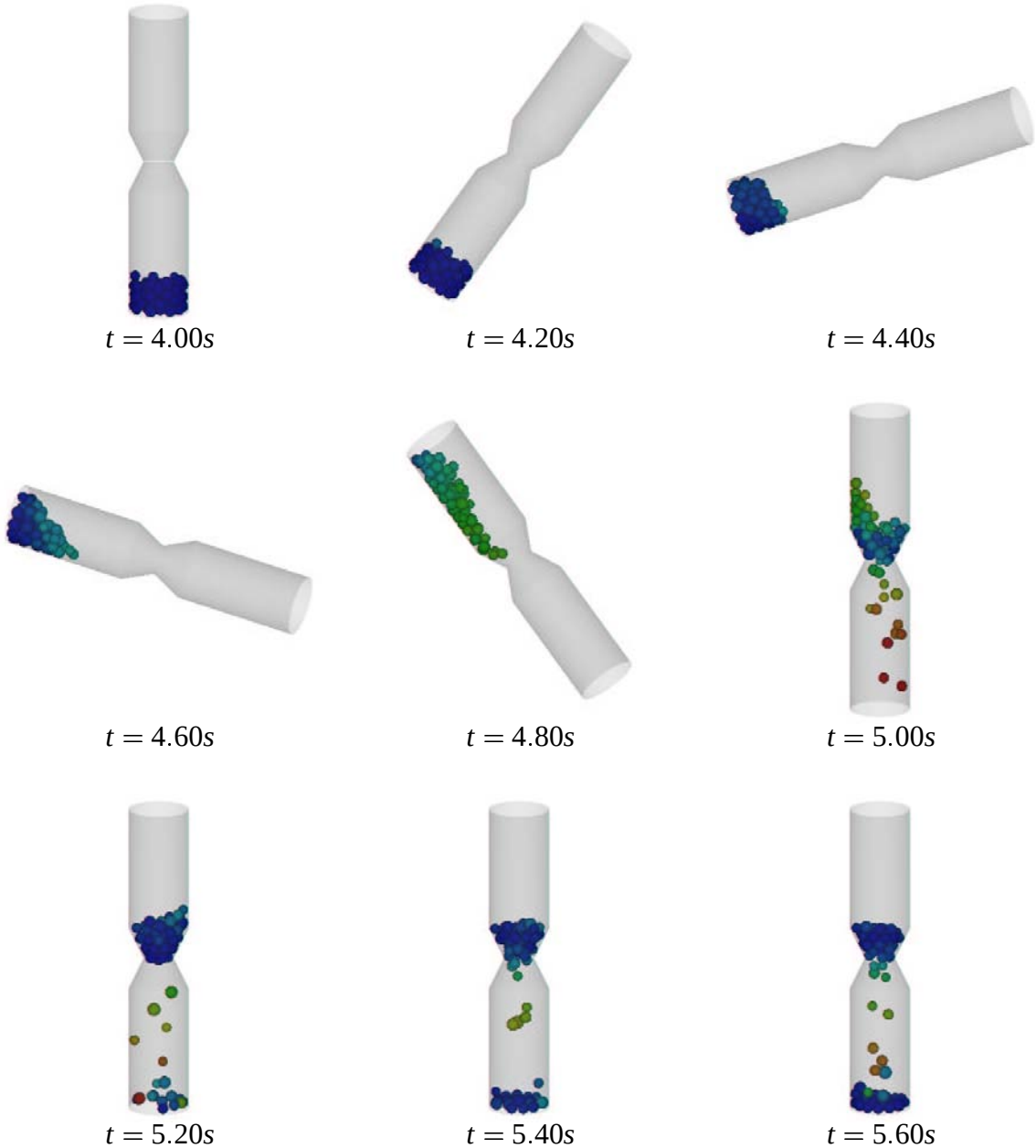


Figure 6.17: Individual snapshots during the slow reversal.

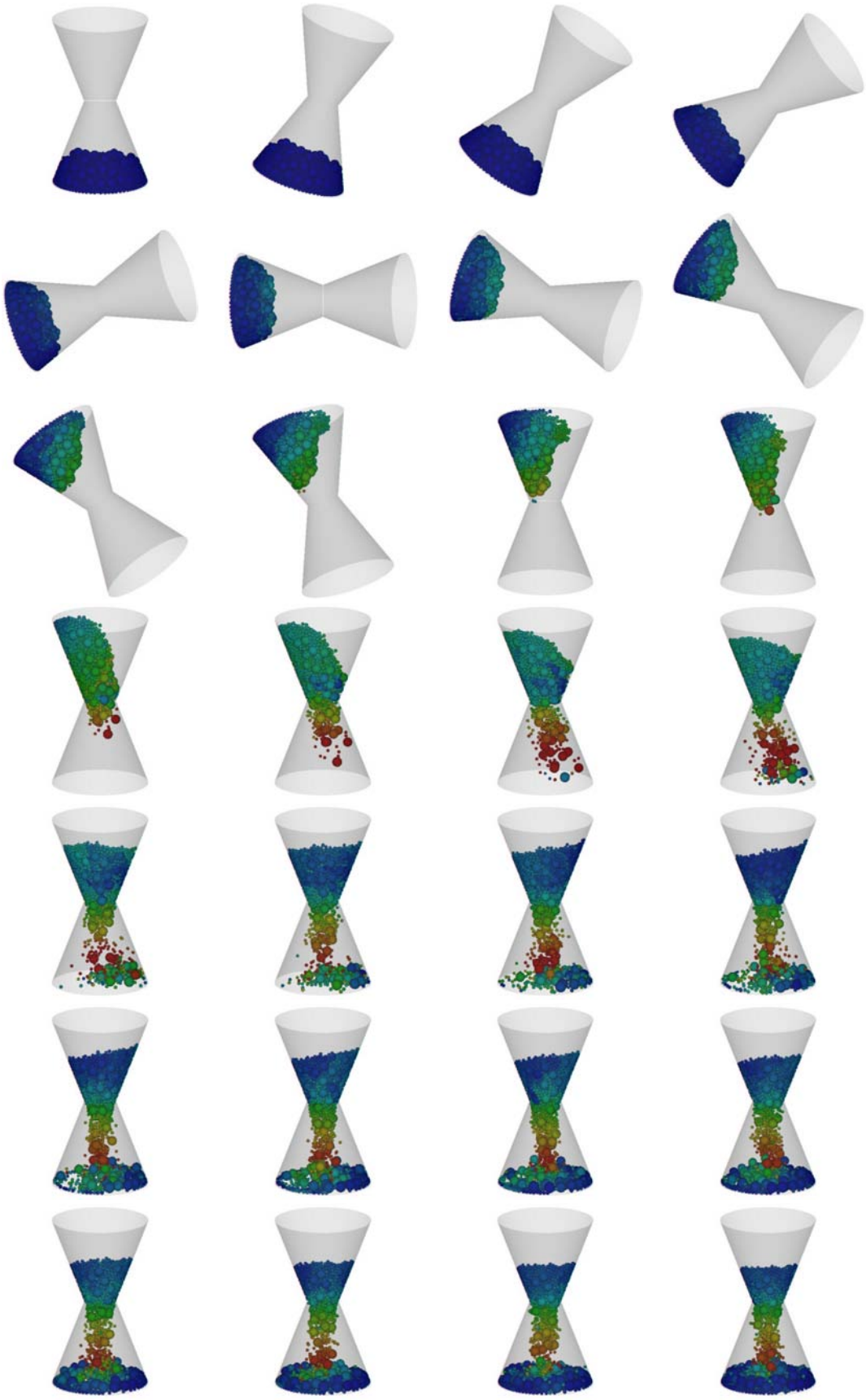


Figure 6.18: Rotating the tall hourglass.

6.4 Clusters of spherical grains

6.4.1 Validation of the concept

We performed two series of simulations with small clusters of spheres glued together by the artificial force described in section 3.5. The grains are placed in a cubic box with energy brought in the system by a vibrating floor (2.5Hz). Ten seconds were simulated, but all cases are divided in two periods: during the first, the grains fall down and rearrange massively according to their shape, while during the second only local movements, mostly periodic and following the vibration are observed. Various snapshots of two cases at different time and from different point of view are given below. The first case comprises 50 short rods built with 15 spheres, the second comprises 125 tetraheda built with 35 spheres.

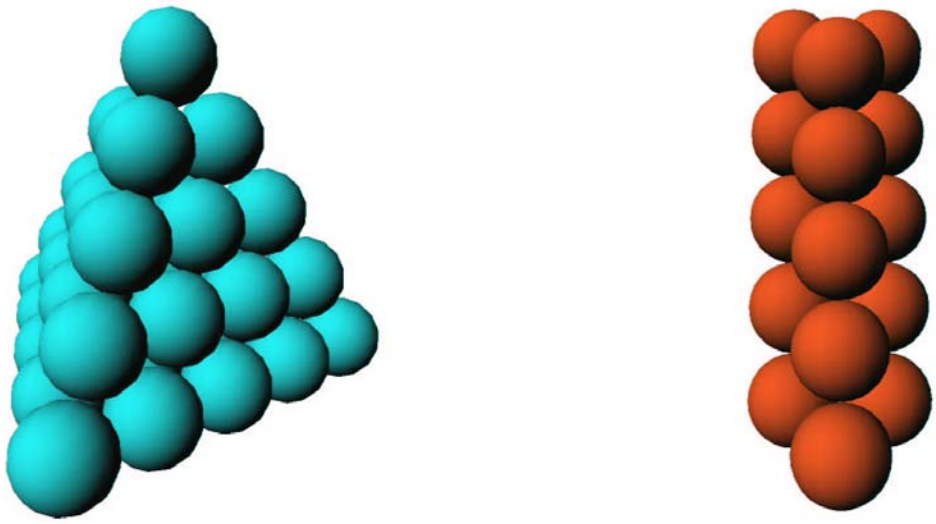
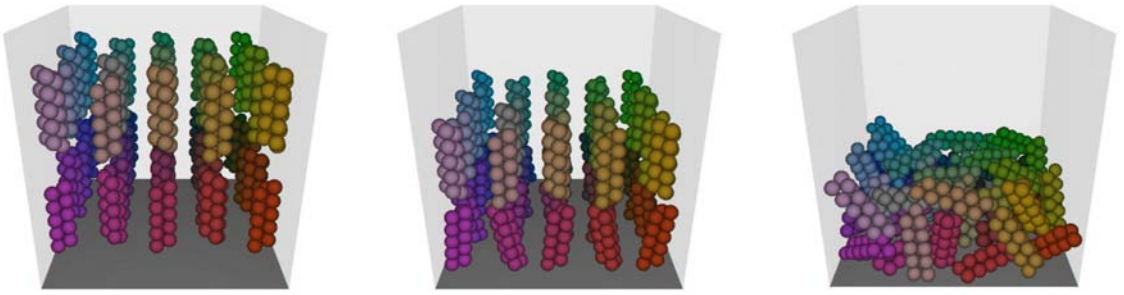
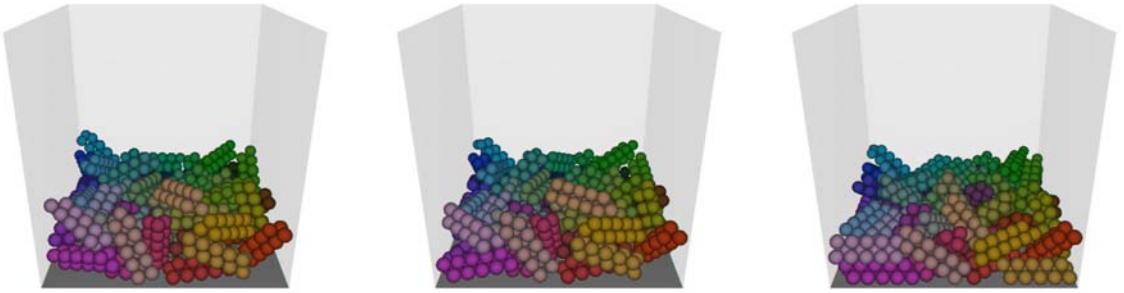


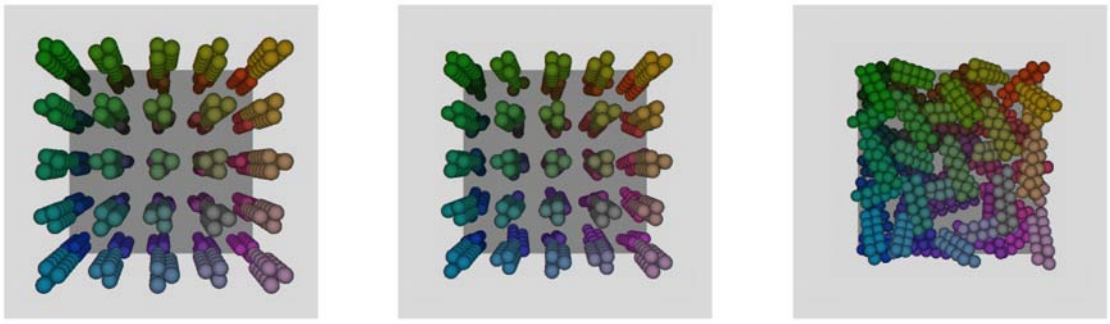
Figure 6.19: The tetrahedron and rod used in the simulations.



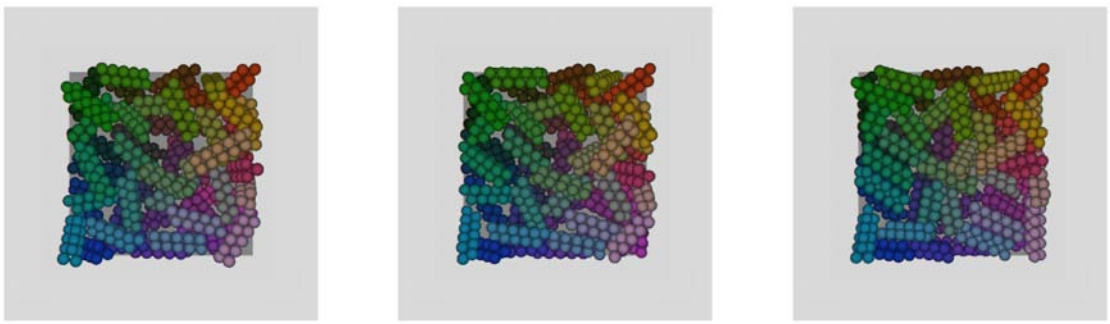
$t = 0.0s$ $t = 0.25s$ $t = 0.5s$



$t = 0.75s$ $t = 1.0s$ $t = 10.0s$

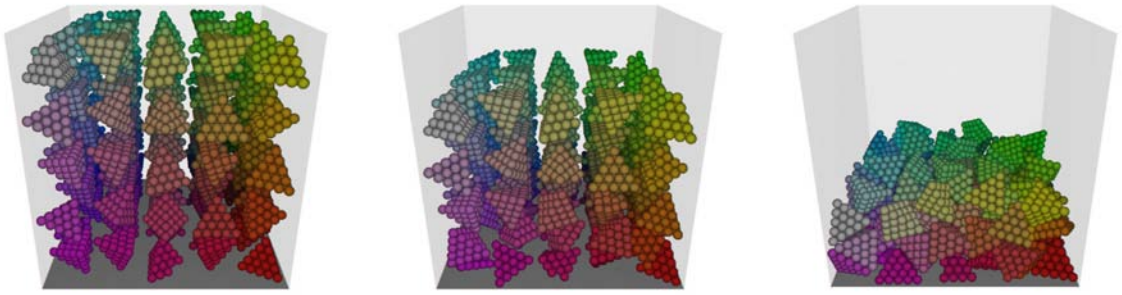


$t = 0.0s$ $t = 0.25s$ $t = 0.5s$

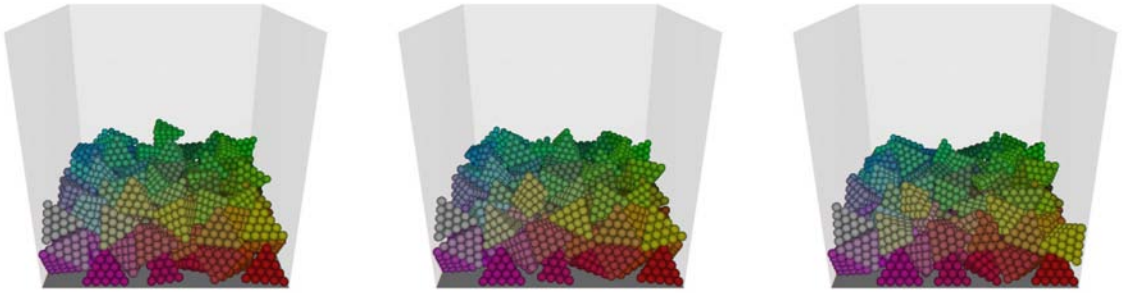


$t = 0.75s$ $t = 1.0s$ $t = 10.0s$

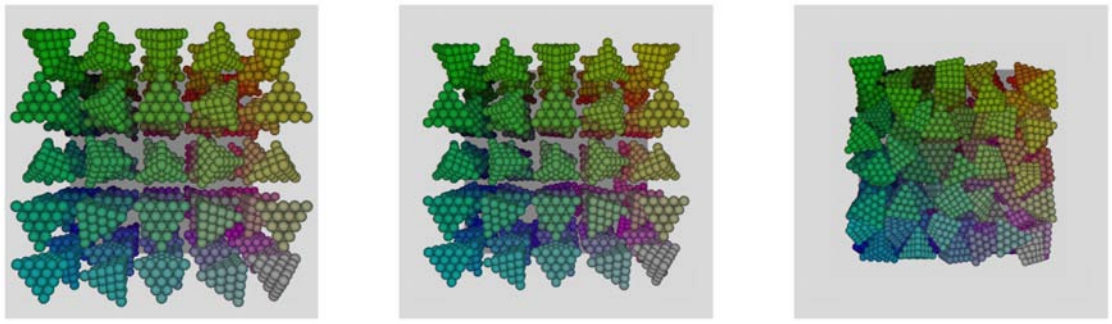
Figure 6.20: Front and top view of 50 short rods.



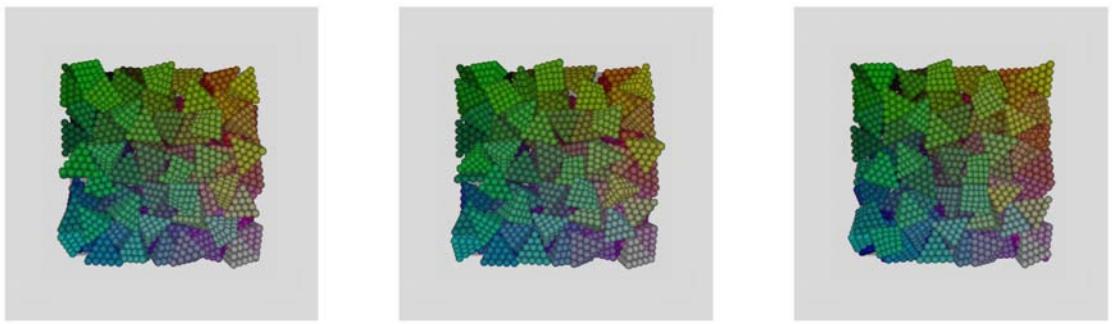
$t = 0.0s$ $t = 0.25s$ $t = 0.5s$



$t = 0.75s$ $t = 1.0s$ $t = 10.0s$



$t = 0.0s$ $t = 0.25s$ $t = 0.5s$



$t = 0.75s$ $t = 1.0s$ $t = 10.0s$

Figure 6.21: Front and top view of 125 tetrahedra.

6.4.2 Calibration of the internal gluing force

Several simulations were made as an attempt to give realistic values to the parameters A, α and β of the force model given by equations (3.27) and (3.29) in section 3.5. Due to the difference between our macroscopic clusters and the atomic forces that inspired the approach, we could not use values traditionally assigned to these parameters.

The simulations involve a vertical cylindrical cluster built with 20 layers of approx. 85 spheres compressed by a piston, as shown in figure 6.22. Once equilibrium is reached where the force exerted by the piston is balanced by the force generated by the deformation of the cylinder, we measure that deformation.

We arbitrarily choosed to fix α and A, in this case $\alpha = 0.2$ and $A = -1500$, and study the influence of β on the behavior of the material. Large values of β are supposed to generate narrow potential and thus hard materials, whereas small values of β widen the area of minimal potential, thus imitating soft materials.

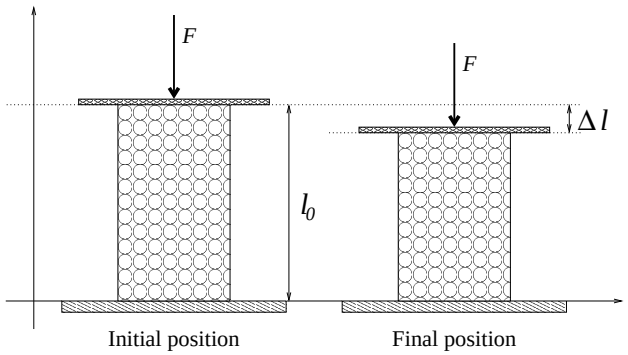


Figure 6.22: A cylindric cluster is compressed by a piston to study the influence of the parameters A, α and β.

The results of 20 simulations are smmarized in table 6.3. For every value of β and of the force F applied by the pistion, the table gives the value of the dimensionless deformation ε defined as

$$\varepsilon = \frac{\Delta l}{l_0} \tag{6.1}$$

where Δl is the penetration depth of the piston and l_0 the length of the cylinder at rest. Elasticity curves, see figure 6.23, confirm that the deformation is proportional to the force exerted, and that higher values of β yield smaller deformations, thus harder materials. (Note that these curves do not pass by the origin, this is due to a small initial gap between the cylinder and the piston.)

Value of β	**Values of ε**				
	10N	**50N**	**100N**	**500N**	**1000N**
1.0	0.003935	0.006073	0.007750	0.018670	0.031611
2.0	0.002726	0.003337	0.004183	0.009463	0.015267
4.0	0.002239	0.002390	0.002726	0.005473	0.008437
8.0	0.002022	0.002080	0.002175	0.003537	0.005200

Table 6.3: Measures of deformation for various values of β and various forces.

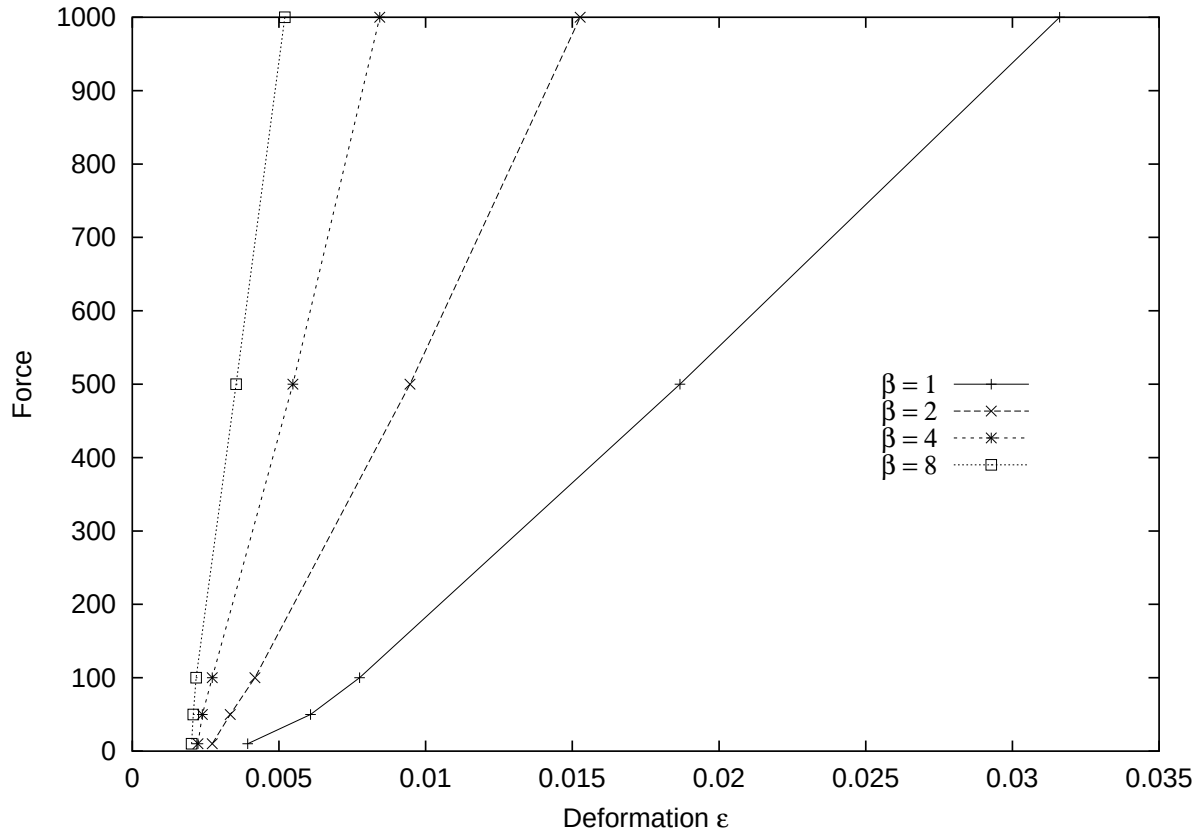


Figure 6.23: Elasticity curves for various values of β and various forces.

6.5 Force visualization

One of the major advantages of computer simulations is to visualize phenomena difficult to observe in a lab experiment. In particular for granular materials, the internal repartition of the contact forces creates arching effects difficult to observe. In 2D, one can use materials such as glass or acetate that polarize light differently in function of the pressure they undergo in order to observe those effects. It is fairly easy to represent such effects in 2D simulations, as shown in figure 6.24 taken from Müller's web page at http://rosowww.epfl.ch/dm/sigma.html.

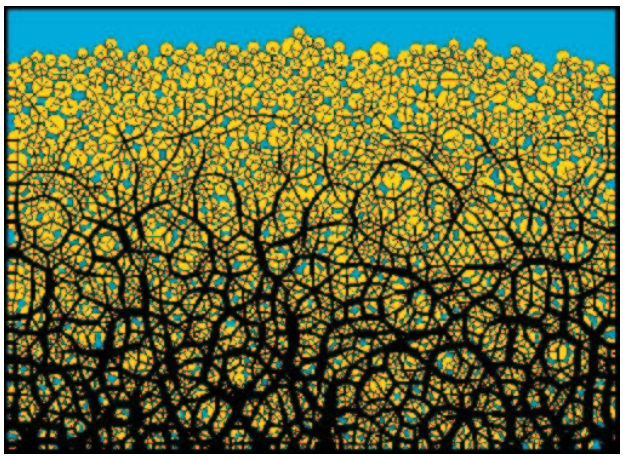
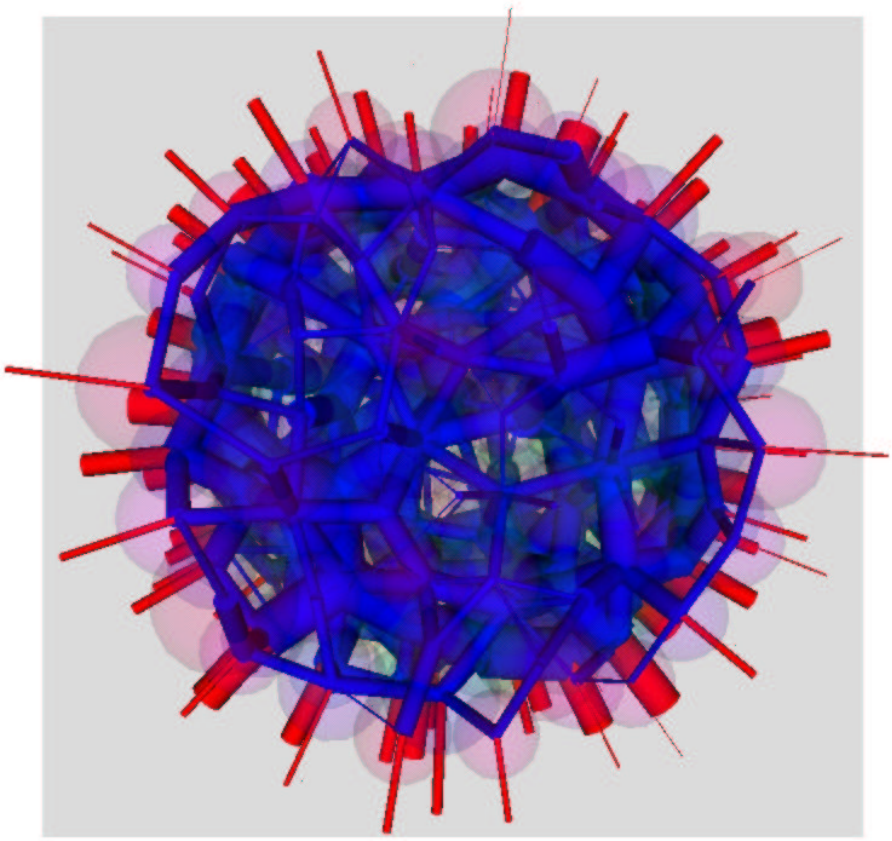


Figure 6.24: Force arches in 2D.

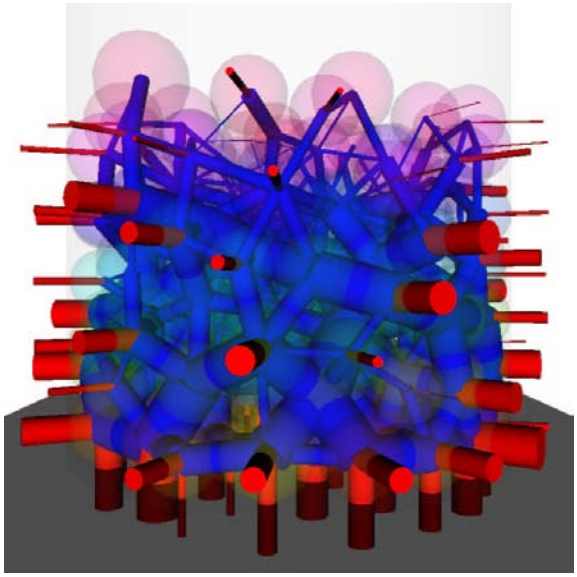
In 3D the technique is similar: draw a cylinder at every contact point to represent the force, with large forces shown by large cylinders. Unfortunately, planar projections of such representations fail to render the volumic information. Furthermore, drawing the grains mostly hides the forces, and not drawing the grains significantly reduces the realism of the representation. With the help of transparency, we managed to produce a few illustrations, but we will leave it up to the reader to decide whether they belong to scientific visualization or to modern art.

A better way to observe these arches is to manipulate interactively 3D models. This can be achieved for example by exporting the simulation in VRML[1] format, and then using specialized software to view these. Full details and several examples are given on our web page at http://rosowww.epfl.ch/jaf/3dwdt/.

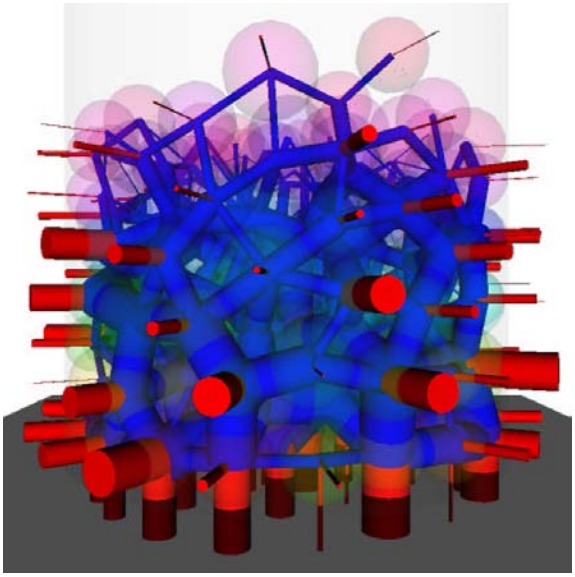
[1] Virtual Reality Modeling Language, see http://www.vrml.org.



Top view



Front view



Side view

Figure 6.25: Visualization of the forces in a packing of 209 grains in a cylinder. The grains are drawn almost transparent, grain-grain forces are drawn in blue and grain-wall forces in red. Because of the gravitation force acting on the grains, the forces are larger at the bottom of the assembly. Some arches are visible.

Life is the process of finding out, too late,
everything that should have been obvious at the time.
– John D. MacDonald

Conclusion

The major theoretical and practical result of this thesis is the successful extension to three-dimensional space of the pioneering work of Müller concerning the efficient detection of collision among spheres with dynamic triangulations.

From a theoretical point of view, we proved that if a weighted 3D Delaunay triangulation built on the centers of spherical grains of any size is maintained to follow the motion of the grains, then the edges of the triangulation identify all potential collisions. Under mild hypotheses, this scheme is efficient as the number of edges is proportional to the number of grains, and the triangulation maintenance can be performed optimally with local operations.

From a practical point of view, we implemented this collision detection scheme in a modular simulation code for 3D granular materials. This code is based on advanced data structures for storing and manipulating the triangulation. Special care is taken for the efficient exact evaluation of geometric predicates as these computations are very sensitive to rounding errors. This code works equally well on standard, single-processor machines and on multi-processor parallel machines offering shared memory access. This yields a significant reduction of computation time for very large problems.

Simulations performed with this method and this tool have confirmed known facts and provided new insights for various phenomena of granular materials. The density of powder packings obtained by mixing grains of various sizes is difficult to determine theoretically. Simulations concerning the influence of the vibrations used to improve the packing, or the local behavior of small and medium grains around large ones allowed us to gain better understanding in this area of uttermost importance to some industrial processes. Other simulations have confirmed the theoretical and experimental assumptions on the unbiasedness of the measures taken by the SOFO sensor for concrete at early and very early age.

By looking at grains in an hourglass we observed several key phenomena of granular flows, namely surface effects, constrained and free flows, arching effects blocking the flow, impact of a flow on a wall, heap formation, etc.

Preliminary experiments were done based on an original idea of gluing spheres together to form non-spherical grains. Some artificial test cases allowed validating the approach, and traditional experiments were replicated in order to fit some parameters of the internal cohesion force.

Many more applications can and hopefully will be studied with the simulation code developed in this thesis. It was in fact designed with reusability in mind, both at the user or programmer level. We seem to have achieved this goal, as several students have successfully used and extended the code during their semester and diploma projects.

Finally, as the conclusion of this thesis is everything but the conclusion of the research effort in this domain, we would like to point out some promising ideas for future developments.

As mentioned in chapter 3, the collision detection used by Müller for 2D polygonal grains will be very hard to extend to 3D polyhedral grains. The constrained triangulation of a general polyhedral domain in three-dimensional space is still object of specific theoretical research, and efficient algorithms to maintain such a triangulation are a prerequisite for its application to collision detection.

In the mean time, more effort can be devoted to the clusters of spherical grains. Aside from allowing a wide range of grain shape, they can also serve as basis for the study of grain deformation in granular materials and in particular the various breakage phenomena. Such simulations will of course find immediate application in the mining industry.

The support for clusters of grains called for the ability to have different force models depending on the nature of the grains involved in the contact. This possibility should be investigated further, maybe in conjunction with adaptive models that are able to react differently according to the situation. This is necessary for applications involving heterogeneous materials, and could be the basis for a first approximation of some wet granular materials where the contact between two grains is influenced by an interstitial fluid.

Whether to account for clusters of spheres, or to study the packing of grains of very different sizes, the need for larger simulations is growing. If 100'000 grains seem now reasonable, some applications will need more. Such large simulations obviously call for more efficient algorithms, codes and computers. In this context, an implementation for distributed memory computers would be welcome, yet not trivial. But aside from the actual computation, the management of those huge data sets requires more and more attention. In particular, the post-processing of the raw simulation results, whether it involves producing density curves, short movies or 3D models of the force arches must be improved. An urgent development in this area is a custom visualization procedure for large sets of particles using the low level but highly optimized OpenGL framework.

All our simulations involved very simple boundary elements: cubes, cylinders, spheres, cones. In order to fulfill the requirements of most industrial cases, complex boundaries must be accounted for. As those geometries often come from CAD software as triangulated volumes, there are nice synergies to be found between the triangulation that detects collisions and the triangulation that describes the objects. In particular, moving boundaries will be easy to manage this way. As with the clusters, this is yet another step in the direction of dynamic, partly constrained 3D triangulations.

Be careful of reading health books,
you might die of a misprint.
– Mark Twain

Bibliography

Algorithmic Solutions Software GmbH (2001). LEDA, a Library of Efficient Data types and Algorithms. http://www.algorithmic-solutions.com/.

Allen, M. P. and Tildesley, D. J. (1987). *Computer simulation of liquids*. Clarendon Press, Oxford.

Alliez, P., Devillers, O., and Snoeyink, J. (1998). Removing degeneracies by perturbing the problem or the world. In *Proc. 10th Canad. Conf. Comput. Geom.* INRIA Research Report 3316, 1997.

Andrade, B. and Reinmann, S. (2001). Simulation d'amas de grains sphériques. Projet de semestre, EPFL-DMA.

Aurenhammer, F. (1987). Power diagrams: properties, algorithms and applications. *SIAM J. Comput.*, 16(1):78–96.

Aurenhammer, F. (1988). Improved algorithms for discs and balls using power diagrams. *J. Algorithms*, 9:151–161.

Bakucz, P., Krause, G., and Stoyan, D. (1999). Force distribution in loaded planar disc systems. In Gaul, L. and Brebbia, C. A., editors, *Computational Methods in Contact Mechanics IV*, pages 273–282. WIT Press, Southampton/Boston.

Barker, G. C. (1994). Computer simulations of granular materials. In Mehta, A., editor, *Granular Matter: An interdisciplinary approach*. Springer-Verlag.

Barriuso, R. and Knies, A. (1994). *SHMEM User's Guide for C (Rev. 2.2)*. Cray Research Inc.

Basch, J., Erickson, J., Guibas, L. J., Hershberger, J., and Zhang, L. (1999). Kinetic collision detection for two simple polygons. In *Proc. 10th ACM-SIAM Sympos. Discrete Algorithms*, pages 102–111.

Basch, J., Guibas, L. J., and Hershberger, J. (1997a). Data structures for mobile data. In *Proc. 8th ACM-SIAM Sympos. Discrete Algorithms*, pages 747–756.

Basch, J., Guibas, L. J., Silverstein, C. D., and Zhang, L. (1997b). A practical evaluation of kinetic data structures. In *Proc. 13th ACM Symposium on Computational Geometry*, pages 388–390. http://graphics.stanford.EDU/~lizhang/interests.html.

Basch, J., Guibas, L. J., and Zhang, L. (1997c). Proximity problems on moving points. In *Proc. 13th ACM Symposium on Computational Geometry*, pages 344–351. http://graphics.stanford.EDU/~lizhang/interests.html.

Berger, R. (1999). Triangulations dynamiques en 3D avec LEDA. Projet de semestre, EPFL-DMA.

Berger, R. (2000). Simulation de la chute d'un bloc rocheux sur un remblai. Projet de semestre, EPFL-DMA.

Bierlaire, M. (2001). A general formulation of the cross-nested logit model. In *Proceedings of the 1st Swiss Transportation Research Conference, Ascona, Switzerland*.

Bircher, H. (1998). Berechnung von theoretischen Schüttdichten. private comm.

Bircher, H., Mathieu, J., and Mäder, P. (1999). Einfluss der Korngrössenverteilung/Kornform von Sprengstoffkristallen auf die rheologishen Parameter von PBX-Sprengstoffen. private comm.

Boissonnat, J.-D. and Yvinec, M. (1995). *Géométrie Algorithmique*. Ediscience. Published in english as Algorithmic Geometry, Cambridge University Press, 1998.

Brönnimann, H., Schirra, S., and Veltkamp, R. (2001). The CGAL Reference Manual. http://www.cgal.org/Manual/.

Bronnimann, H. and Yvinec, M. (1997). Efficient exact evaluation of signs of determinants. In *Proc. 13th Annu. ACM Sympos. Comput. Geom.*, pages 166–173.

Carreira, J., Silva, L. M., Silva, J. G., and Chapple, S. (1995). Implementing Virtual Shared Memories on MPI. In *MPI Developers Conference*, University of Notre Dame.

Chandra, R., Chen, D., Cox, R., Maydan, D. E., Nedeljkovic, N., and Anderson, J. M. (1997). Data distribution support on distributed shared memory multiprocessors. In *Proc. SIGPLAN 97*.

Chung, K. and Weng, W. (1996). Quick collision detection of polytopes in virtual environments. In *Proc. 3rd ACM Sympos. Virtual Reality Software and Technology*, pages 125–132.

Cignoni, P., Laforenzy, D., Montani, C., Perego, R., and Scopigno, R. (1995). Evaluation of parallelization strategies for an incremental Delaunay triangulator in E3. *Concurrency: Practice and Experience*, 7(1):61–80.

Cignoni, P., Montani, C., Perego, R., and Scopigno, R. (1993). Parallel 3D Delaunay triangulation. *Computer Graphics Forum*, 12(3):129–142.

Claudin, P. (1999). *La physique des tas de sable*. EDP Sciences.

Cleary, P. W. (1998). Discrete element modelling of industrial granular flow applications. *TASK Quarterly*, 2(3):385–415.

Cleary, P. W. and Sawley, M. L. (1999). Three-dimensional modelling of industrial granular flows. In *Second International Conference on CFD in the Minerals and Process Industries*. CSIRO, Melbourne, Australia.

Cundall, P. A. (1988). Formulation of a three-dimensional distinct-element model — Part I. A scheme to detect and represent contacts in a system composed of many polyhedral blocks. *Int. J. Rock Mech. Min. Sci. & Geomech. Abstr.*, 25(3).

Cundall, P. A. and Strack, O. D. L. (1979). A discrete numerical model for granular assemblies. *Géotechnique*, 29(1).

Dai, Y. and Sutou, A. (2000). A study of the global optimization approach to spherical packing problems. Technical Report B-361, Tokyo Institue of Technology. http://www.is.titech.ac.jp/research/research-report/B/.

Devillers, O., Golin, M., Kedem, K., and Schirra, S. (1996). Queries on Voronoi diagrams of moving points. *Comput. Geom. Theory Appl.*, 6:315–327.

Dobkin, D. P. and Laszlo, M. J. (1989). Primitives for the manipulation of three-dimensional subdivisions. *Algorithmica*, 4:3–32.

Dos Reis, G. (1998). Vers une nouvelle approche du calcul scientifique en C++. Technical Report RR-3362, INRIA. http://www.inria.fr/rrrt/rr-3362.html.

Drake, T. G. (1991). Granular flow: physical experiments and their implications for microstructural theories. *J. Fluids. Mech.*, 12(225).

Duran, J. (2000). *Sands, Powders, and Grains: An introduction to the Physics of Granular Materials*. Springer.

Edelsbrunner, H. (1987). *Algorithms in Combinatorial Geometry*. Springer-Verlag.

Edelsbrunner, H. and Mücke, E. P. (1990). Simulation of simplicity: a technique to cope with degenerate cases in geometric algorithms. *ACM Trans. Graph.*, 6:66–104.

Edelsbrunner, H. and Shah, N. R. (1996). Incremental topological flipping works for regular triangulations. *Algorithmica*, 15:223–241.

Erickson, J., Guibas, L. J., Stolfi, J., and Zhang, L. (1999). Separation-sensitive collision detection for convex objects. In *Proc. 10th ACM-SIAM Sympos. Discrete Algorithms*, pages 327–336.

Feiner, S., Foley, J., Dam, A. V., and Hughes, J. F. (1990). *Fundamentals of interactive computer graphics*. Addison Wesley, USA, 2nd edition.

Ferrez, J.-A., Fukuda, K., and Liebling, T. M. (1998a). Parallel computation of the diameter of a graph. In Schaeffer, J., editor, *High Performance Computing Systems and Applications*, pages 283–296. Kluwer Academic Publishers.

Ferrez, J.-A., Fukuda, K., and Liebling, T. M. (1998b). Parallel implementation of graph diameter algorithms. *EPFL Supercomputing Review*, 10. Online at http://sawww.epfl.ch/SIC/SA/publications/SCR98/scr10-page3.html.

Ferrez, J.-A., Glisic, B., and Liebling, T. M. (2000a). Distinct element simulation of the SOFO sensor for concrete at early and very early age. Technical Report RO001002.

Ferrez, J.-A. and Liebling, T. M. (2000). Using dynamic triangulations in distinct element simulations. In Deville, M. and Owens, R., editors, *Proceedings of the 16th IMACS World Congress*.

Ferrez, J.-A. and Liebling, T. M. (2001a). Dynamic triangulations for efficient collision detection among spheres with applications in granular media simulations. *Submitted to Phil. Mag. B*.

Ferrez, J.-A. and Liebling, T. M. (2001b). An environment for granular media simulations based on dynamic 3D weighted Delaunay triangulations. *in preparation*.

Ferrez, J.-A., Müller, D., and Liebling, T. M. (1996). Parallel implementation of a distinct element method for granular media simulation on the Cray T3D. *EPFL Supercomputing Review*, 8. Online at http://sawww.epfl.ch/SIC/SA/publications/SCR96/scr8-page4.html.

Ferrez, J.-A., Müller, D., and Liebling, T. M. (2000b). Dynamic triangulations for granular media simulations. In Mecke, K. R. and Stoyan, D., editors, *Statistical Physics and Spatial Statistics*, Lecture Notes in Physics. Springer.

Ferrez, J.-A., Pournin, L., and Liebling, T. M. (1999). A simulation environment for packings of 3D spheres. Final Report, Simboules project.

Ferrez, J.-A., Pournin, L., and Liebling, T. M. (2000c). Distinct element computer simulations for optimal packings of 3D spheres. Final Report, Simboules project, second year.

Foester, S. F., Louge, M. Y., Chang, H., and Allia, K. (1994). Measurement of the collision properties of small spheres. *Phys. Fluids*, 6(1108).

Folly, P. (2000). Experimental measures of densities with steel balls. private comm.

Fortune, S. and Van Wyk, C. J. (1993). Efficient exact arithmetic for computational geometry. In *Ninth Annual Symposium on Computational Geometry*, pages 163–172.

Fukuda, K. (2001). On quadratic 3d triangulations. private comm.

Ghaboussi, J. and Barbosa, R. (1990). Three-dimensionnal discrete element method for granular materials. *Int J. for Numerical and Analytical Methods in Geomechanics*, 14(451).

Glisic, B. (2000a). *Fiber Optic Sensors and Behaviour in Concrete at Early Age*. Thèse N° 2168, EPFL.

Glisic, B. (2000b). Limitation de la fissuration au jeune age du béton dans des structures hybrides. Research report, MCS, IMAC and ICOM, EPFL, Lausanne, Switzerland.

Goldberg, D. (1991). What every computer scientist should know about floating-point arithmetic. *ACM Computing Surveys*, 23(1):5–48.

Guibas, L., Mitchell, J. S. B., and Roos, T. (1991). Voronoi diagrams of moving points in the plane. In *Proc. 17th Internat. Workshop Graph-Theoret. Concepts Comput. Sci.*, volume volume 570 of Lecture Notes in Computer Science, pages 113–125. Springer-Verlag.

Guibas, L. and Stolfi, J. (1985). Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams. *ACM Transactions on Graphics*, 4(2):74–123.

Guibas, L. J., Knuth, D. E., and Sharir, M. (1992). Randomized incremental construction of Delaunay and Voronoi diagrams. *Algorithmica*, 7:381–413.

Guibas, L. J. and Zhang, L. (1998). Euclidean proximity and power diagrams. In *Proc. 10th Canadian Conference on Computational Geometry*. http://graphics.stanford.EDU/~lizhang/interests.html.

Gustafson, J., Snell, Q., Amit, S., Heller, D., and Todi, R. (1999). The HINT benchmark. http://www.scl.ameslab.gov/HINT/.

Hales, T. C. (1998). The Kepler conjecture. Web page with an overview of the proof and related papers, http://www.math.lsa.umich.edu/~hales/countdown/.

Hoomans, B. P. B., Kuipers, J. A. M., Briels, W. J., and Swaaij, W. P. M. V. (1996). Discrete particle simulation of bubble and slug formation in a two-dimensional gas-fluidised bed: a hard-sphere approach. *Chemical Engineering Science*, 51(1):99–118.

Hopcroft, J. E., Schwartz, J. T., and Sharir, M. (1983). Efficient detection of intersections among spheres. *Intl. J. Robotics Research*, 2(4):77–80.

Imai, H., Iri, M., and Murota, K. (1985). Voronoi diagram in the Laguerre geometry and its applications. *SIAM J Computing*, 14(1):93–105.

Inaudi, D. (1997). *Fiber Optic Sensor Network for the Monitoring of Civil Structures*. Thèse N° 1612, EPFL.

Indermitte, C. (1995). *Modélisation et simulation de la croissance d'un mycélium*. Thèse N° 1404, EPFL.

Jenkins, J. T. and Savage, S. B. (1983). A theory for the rapid flow of identical, smooth, nearly elastic, spherical particles. *J Fluid Mech*, 130:187–202.

Joe, B. (1989). Three-dimensional triangulations from local transformations. *SIAM J. Sci. Stat. Comput.*, 10(4):718–741.

Joe, B. (1991). Construction of three-dimensional Delaunay triangulations using local transformations. *Computer Aided Geometric Design*, 8:123–142.

Joe, B. (1992). Three-dimensional boundary-constrained triangulations. In Houstis, E. N. and Rice, J. R., editors, *Artificial Intelligence, Expert Systems, and Symbolic Computing*, pages 215–222. Elsevier Science Publishers. ftp://ftp.cs.ualberta.ca/pub/geompack/Papers/Joe92.ps.Z.

Joe, B. (1993). Construction of k-dimensional Delaunay triangulations using local transformations. *SIAM J. Sci. Comput.*, 14(6):1415–1436.

Joe, B. (1995). Construction of three-dimensional improved-quality triangulations using local transformations. *SIAM J. Sci. Comput.*, 16:1929–1307.

Johnson, K. (1985). *Contact Mechanics*. Cambridge Univ. Press, New York.

Kuonen, P. and Gruber, R. (1999). Parallel computer architectures for commodity computing and the Swiss-T1 machine. *EPFL Supercomputing Review*, 11. Online at http://sawww.epfl.ch/SIC/SA/publications/SCR99/scr11-page3.html.

Kurz, W., Mercier, J. P., and Zambelli, G. (1991). *Introduction à la Science des Matériaux*, volume 1 of *Traité des Matériaux*. Presses Polythechniques et Universitaires Romandes, 2ème edition.

Kuwabara, G. and Kono, K. (1987). Restitution coefficient in a collision between two spheres. *Jap. J. Appl. Phys.*, 26(1230).

Labous, L., Rosato, A. D., and Dave, R. N. (1997). Measurements of collisional properties of spheres using high-speed video analysis. *Phys. Rev. E*, 56(5717).

Lamarche, F. and Leroy, C. (1990). Evaluation of the volume of intersection of a sphere with a cylinder by elliptic integrals. *Computer Phys. Comm.*, 59:359–369.

Legrand, C. and Wirquin, E. (1994). First developments of strength in a microconcrete. In *Thermal cracking in concrete at early age, RILEM International Symposium, Munich*, pages 89–100.

Leutwyler, K. (1998). Stack 'em tight. Scientific American. Online at http://www.sciam.com/explorations/1998/091498sphere/.

Levine, J., Mason, T., and Brown, D. (1992). *lex & yacc*. O'Reilly, 2nd edition.

Liebling, T. M., Mocellin, A., Telley, H., Righetti, F., Clémençon, H., and Indermitte, C. (1992). Nouvelles approches dans la modélisation et simulation de processus de croissance en science des matériaux et en biologie. *Cahiers du CERO*, 34:117–138.

Luding, S., Clément, E., Blumen, A., Rajchenbach, J., and Duran, J. (1994). Anomalous energy dissipation in molecular-dynamics simulations of grains: The "detachment" effect. *Phys. Rev. E*, 50(4113).

Magnier, S.-A. and Donzé, F. V. (1998). Numerical simulation of impacts using a discrete element method. *Mech. Cohes.-frict. Mater.*, 3.

Mantyla, M. (1988). *An introduction to solid modeling*. Computer Science Press, USA.

Matuttis, H.-G., Luding, S., and Herrmann, H. J. (2000). Discrete element methods for the simulation of dense packings and heaps made of spherical and non-spherical particles. *Powder Technology*, 109(1-3):278–292.

Mücke, E. (1993). *Shapes and Implementations in Three-Dimensional Geometry*. PhD thesis, Department of Computer Science, University of Illinois at Urbana-Champaign. Technical Report UIUCDCS-R-93-1836, ftp://cs.uiuc.edu/pub/TechReports/UIUCDCS-R-93-1836.ps.Z.

McNamara, S. and Young, W. R. (1992). Inelastic collapse and clumping in a one-dimensional granular medium. *Phys. Fluids A*, 4(496).

Mehlhorn, K. and Näher, S. (1999). *LEDA*. Cambridge University Press.

Meichtry, E. (2001). Simulation d'amas de grains sphériques. Projet de semestre, EPFL-DMA.

Meichtry, E. and Paroz, S. (2000). Animations interactives de triangulations 3D. Projet de semestre, EPFL-DMA.

Mirtich, B. (1997). V-clip: Fast and robust polyedral collision detection. Technical Report TR-97-05, Mitsubishi Electrical Research Laboratory.

Mirtich, B. and Canny, J. (1995). *Impulse-based dynamic simulation, The agorithmic Foundations of robotics*. A. K. Peters.

Müller, D. (1996a). *Techniques informatiques efficaces pour la simulation de mileux granulaires par des méthodes d'éléments distincts*. Thèse N° 1545, EPFL.

Müller, D. (1996b). Web page with example simulations of granular media. http://rosowww.epfl.ch/dm/sigma.html.

Müller, D. (1998). Mieux comprendre les milieux granulaires. *IAS Bulletin*, 9.

Müller, D. and Liebling, T. M. (1995). Detection of collisions of polygons by using a triangulation. In et al., M. R., editor, *Contact Mechanics*, pages 369–372. Plenum Publishing Corporation, New York.

Monaghan, J. J. (1992). Smoothed particle hydrodynamics. *Annu. Rev. Astron. Astrophys.*, 30:543–574.

Montani Stoffel, S. (1998). *Sollicitation dynamique de la couverture des galeries de protection lors de chutes de blocs*. Thèse N° 1899, EPFL.

Muguruma, Y., Tanaka, T., and Tsuji, Y. (2000). Numerical simulation of particulate flow with liquid bridge between particles (simulation of centrifugal tumbling granulator). *Powder Technology*, 109:49–57.

Ng, T.-T. and Dorby, R. (1994). Numerical simulations of monotonic and cyclic loading of granular soil. *J. of Geotechnical Engineering*, 120(338).

Nonat, A. and Mutin, J. C. (1994). From hydration to setting. In *Thermal cracking in concrete at early age, RILEM International Symposium, Munich*, pages 171–191.

O'Connor, R. M. (1996). *A distributed discrete element modeling environment - Algorithms, implementation and applications*. PhD thesis, MIT.

Pournin, L. (1999). Triangulations dynamiques pour la simulation de milieux granulaires. Projet de diplôme, EPFL-DMA.

Pournin, L., Liebling, T. M., and Mocellin, A. (2001). A new molecular dynamics force model for better control of energy dissipation in DEM simulations of dense granular media. *to appear in Phys. Rev. E*.

Priest, D. M. (1991). Algorithms for arbitrary precision floating point arithmetic. In Kornerup, P. and Matula, D., editors, *Tenth Symposium on Computer Arithmetic*, pages 132–143. IEEE Computer Society Press.

Pöschel, T. and Buchholtz, V. (1995). Molecular dynamics of arbitrary shaped granular particles. *J. Phys. I*, 5(1431).

Radjai, F., Jean, M., Moreau, J.-J., and Roux, S. (1996). Force distributions in dense two-dimensional granular systems. *Phys. Rev. Lett.*, 77(2):274–277.

Righetti, F. (1992). *Modélisation 3D d'amas polycristallins et méthodologie et méthodologie d'analyse de leurs images microscopiques*. Thèse N° 1016, EPFL.

Ristow, G. H. (2000). *Pattern formation in Granular Materials*. Springer Tracts in Modern Physics. Springer.

Rogers, D. (1985). *Procedural Elements for Computer Graphics*. McGraw Hill.

Sadd, M. H., Tai, Q., and Shukla, A. (1993). Contact law effects on wave propagation in particulate materials using distinct element modeling. *Int. J. Non-Linear Mechanics*, 28(251).

Samet, H. (1984). The quadtree and related hierarchical data structures. *ACM Computing Surveys*, 16:187–260.

Sawley, M. L. and Cleary, P. W. (1999). A parallel discrete element method for industrial granular flow simulations. *EPFL Supercomputing Review*, 8. Online at http://www.epfl.ch/SIC/SA/publications/SCR99/scr11-page23.html.

Schäfer, J., Dippel, S., and Wolf, D. E. (1996). Force schemes in simulations of granular materials. *J. Phys. I*, 6(5).

Shewchuk, J. R. (1996). Robust adaptive floating-point geometric predicates. In *Proceedings of the Twelfth Annual Symposium on Computational Geometry*. ACM. http://www.cs.cmu.edu/~quake/robust.html.

Shewchuk, J. R. (1997). Adaptive precision floating-point arithmetic and fast robust geometric predicates. *Discrete & Computational Geometry*, 18:305–363. http://www.cs.cmu.edu/~quake/robust.html.

Sigurgeirsson, H., Stuart, A., and Wan, W.-L. (2000). Collision detection for particles in a flow. http://www.maths.warwick.ac.uk/~stuart/hersir2.ps.

Sondergaard, R., Caney, K., and Brennen, C. E. (1990). Measurements of solid spheres bouncing off flat plates. *ASME J. Appl. Mech.*, 57(694).

Springenschmid, R., editor (1994). *Thermal cracking in concrete at early age, RILEM International Symposium, Munich*.

Stroustrup, B. (1997). *The C++ Programming Language*. Addison Wesley, 3rd edition.

Sugihara, K., Okabe, A., and Boots, B. (1992). *Spatial Tesselations — Concepts and Applications of Voronoi Diagrams*. John Wiley.

Tazawa, E. and Iida, K. (1983). Mechanism of thermal stress generation due to hydration heat of concrete. *Transactions of the Japan Concrete Institute*, 5:119–126.

Tejchman, J. and Gudehus, G. (1993). *Powder Technology*, 76(201).

Telley, H. (1989). *Modélisation et simulation bidimensionnelle de la croissance des polycristaux*. Thèse N° 780, EPFL.

Telley, H., Liebling, T. M., and Mocellin, A. (1996a). The Laguerre model of grain growth in two dimensions: Part I. Cellular structures viewed as dynamical Laguerre tesselations. *Phil. Mag. B*, 73(3):395–408.

Telley, H., Liebling, T. M., and Mocellin, A. (1996b). The Laguerre model of grain growth in two dimensions: Part II. Examples of coarsening simulations. *Phil. Mag. B*, 73(3):409–427.

Teng, Y. A., Sullivan, F., Beichl, I., and Puppo, E. (1993). A data-parallel algorithm for the three-dimensional Delaunay triangulation and its implementation. In *Proceedings of Supercomputing '93, Portland, Oregon*, pages 112–121.

The CGAL Team (2001). CGAL, Computational Geometry Algorithm Library. http://www.cgal.org/.

Walton, O. R. and Braun, R. L. (1986). Viscosity, granular-temperature, and stress calculations for shearing assemblies of inelastic, frictional discs. *J. of Rheology*, 30(949).

Weibel, C. (2000). Parallélisation d'un algorithme de simulation de milieux granulaires. Projet de diplôme, EPFL-DMA.

Wenzel, O. and Bicanic, N. (1993). A quad tree based contact detection algorithm. In *Proceedings of the 2nd international conference on discrete element methods (DEM), MIT*. IESL Publications.

Xue, X. (1995). *Laguerre models for grain growth*. Thèse N° 1466, EPFL.

Xue, X., Righetti, F., Telley, H., Liebling, T. M., and Mocellin, A. (1997). The Laguerre model for grain growth in three dimensions. *Phil. Mag. B*, 75(4):567–585.

Jean-Albert Ferrez rosowww.epfl.ch/jaf
Born March 26, 1971 jaf@verbier.ch

EDUCATION

1990 – **École Polytechnique Fédérale de Lausanne (EPFL)**: www.epfl.ch
1999: Postgraduate course "*Operations Research and Statistics*".
1997: Postgraduate course "*Methods and Applications on Parallel Computers*".
1995: Teaching certificate in applied mathematics.
1995: Diploma in mathematical engineering.

1985 – 1990 **Secondary school at the Lycée Collège des Creusets, Sion**: Maturity C (scientific).

PROFESSIONAL EXPERIENCE

1995 – **EPFL – Department of Mathematics, Lausanne**: rosowww.epfl.ch
Assistant of Prof. Th. M. Liebling, chair of Operations Research

1993 – 1995 **HotSoft Development, Verbier**: www.hotsoft.com
Consulting and development, Hotel Management Software

VARIOUS

Languages: Fluent in French and English, good knowledge of German, notions of Swiss-German and Italian.
Substitute Deputy in the Parliament of Canton du Valais. www.vs.ch
Vice-president of the local Linux User Group (GULL). www.linux-gull.ch