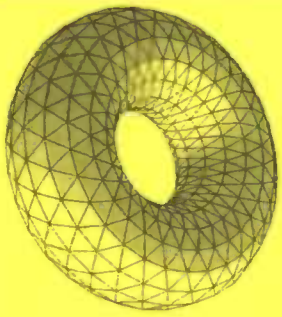
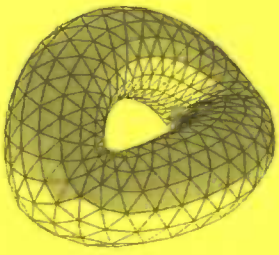
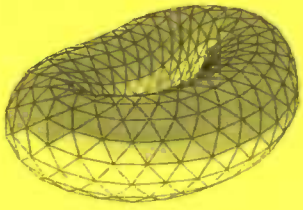
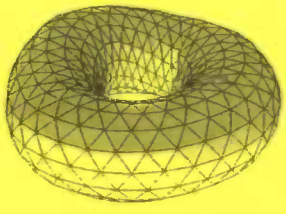
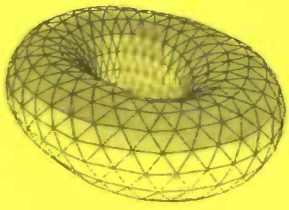
Rijksuniversiteit Groningen











Real-time model experiments for deformable object simulation

Final thesis Computational Science and Visualisation
Egbert van der Es

Supervisor
Henk Bekker

June 28, 2006

RuG

Abstract

In the field of computer graphics deformable object models have not really conquered the real-time industry. Despite decades of research it seems that the models that are currently available do not appeal to game developers and other virtual environment specialists very much. Deformable body simulation requires a significant amount of processing power and there are limitations on the range of materials that can be handled and the amount of 'abuse' it can take. An important field where the deformable object simulation is of big interest is virtual surgery, which offers virtual medical training. Though workers in this field are willing to invest in powerful hardware, current models leave much to be desired for there as well.

With an open mind we started to think of possibly new models that would offer an improvement of these issues for linear elastic material simulation. We aimed for a model that could produce reasonably accurate real world approximations based on real world material parameters. Preferably such a model has an intuitive design and runs at or close to real-time rates. We did not look deep into features like the support for cutting or fracturing the body during simulation at this point.

Three models are discussed in this paper, in the order by which we have studied them. The first two are strongly relying on the traditional mass-spring principles. We started of with the CMPM, a model that a fellow student worked on under the same supervisor. We increased the performance by using another type of numerical integrator and analysed the driving mechanism. The second model was an experiment to see if we could indirectly produce correct shear response. The third model is not directly based on traditional mass-spring systems. The dynamics of the simulated body is the result of the behaviour of separately modeled tetrahedra, connected to each other at their vertices. Though not tested rigorously, experiments with simple shapes produced responses that so far are close to what one would expect from pure linear materials.

Preceding the three model descriptions are two chapters on continuum mechanics and computational physics. Both chapters are added to provide a reference to the fundamentals which these types of simulations are based on.

Contents

1 Introduction

In the last decades computer graphics and real world simulation have made a big flight. The last year deformable bodies have had special attention since more computers have the capacity to work with models in higher resolutions. Fields of work where the simulation of deformable media is put to good use can be found in the entertainment sector, virtual reality environments, construction testing or any other place where realistic physics in a virtual world are asked for.

Both computer simulation and computer graphics have benefitted from developments in hardware and software. New and more powerful hardware has given developers the opportunity to run the same programs in less time, whereas new and more advanced algorithms offered more realistic results. The computer graphics field moved from monochrome pixel drawing to drawing high resolution texturized surfaces. The field of computational physics started with the simulation of point masses, moved to solid objects and from there to deformable objects.

However, the step from solid to deformable volumes has proven to be a difficult one. Though it has been successfully deployed for special effects or crash test simulators one has to consider that these are most often offline simulations, supported by large budgets for high performance hardware. For potential applications of deformable media simulation at home, schools or work there is plenty of room for improvement. Most methods have such high processing requirements that for useful application they are yet beyond the capabilities of low cost hardware. The cheaper algorithms are often suffering from limited realism and flexibility.

Recreating the dynamics of deformable bodies into a computer is not easy. What makes flexible objects hard to recreate on the computer is the large amount of interactions taking place inside the volume. Every piece of matter is directly or indirectly linked to all other matter which influenced by external forces, may be pulled or pushed in any direction. The continuum mechanics branch of mathematical physics has worked on a mathematical foundation to understand these processes and model them in the form of partial differential equations. The field of numerical mathematics tries to translate these partial differential equations to a form computers can cope with so that the results are approximating the real life experience.

Different models for deformable media simulation have been developed over time. So far the ultimate method, which outperforms any other model in speed and approximation accuracy has yet to be discovered. This being unlikely to happen at all, many different approaches have earned their place in this field each having its upsides and downsides when it comes to speed, accuracy and the range of material types it can simulate. This last point is non-trivial since not every material in the real world behaves in the same way. Elasticity, density, plasticity and compressibility are some of the characteristics by which materials can be distinguished. Non-isotropic materials behave differently when the same forces are applied in different directions. In compound materials these characteristics may vary over the volume resulting in inhomogeneity.

1.1 Model types

Sarah Gibson's survey [GM97] about deformable models is almost a decade old, but despite that much of the information on different model types is still true today. Here we'll present an overview of model types described in Sarah's paper, completed with the mentioning of some more recent models.

Non-physics deformable models are common in 3d modeling applications to construct organic shapes. In one type of model surfaces represented by splines are controlled by control points which cause deformation when moved. Free form deformation is a model where the space in which a body resides is deformed, taking the body with it. An example of this technique is tweening in computer games, where the limbs of virtual characters are animated this way. Both approaches offer a large amount of control so that by endless tuning virtually everything is possible. Thought there are tools to speed up interaction, much has to be done by hand.

Getting physically realistic results is a hard and time consuming process and requires sharp observational skills to get it done right.

Physics based models apply the knowledge we have about the dynamics in materials. One of the simplest and best known way to do this is by point masses that are connected to each other by one-dimensional springs. A disadvantage of this type of model is that it requires many nodes to get good results. Also the control on the material behaviour is often limited to adjusting spring and damping constants which does not allow for many different types of material. Many variations on the mass-spring model have been produced to address its shortcomings. Mass-spring models are capable of providing real-time interaction.

Continuum models aim to reduce the potential energy of a material to zero, so that applied forces and resisting material stresses are balanced. The function that describes the potential energy inside the body can be considered to be a boundary value problem. With the finite element methods (FEM) one can find an piecewise linear or higher order approximation for the solution of boundary value problems. The principle behind FEM is that an approximation can be expressed as a combination of a finite set of functions, which may also be referred to as elements. These elements are joined by node points inside the body. The approximation to the solution provides movement constraints for the node points. FEM delivers accurate results for many types of materials but is often too computationally expensive to be used in real time graphics. It is often used for construction analysis, offline and on high performance hardware. In [JP99] James and Pai suggest to use a boundary element method (BEM), which has much in common with FEM, but does not need the body to be divided in smaller element. By taking advantage of similarity between changing external forces they were able to achieve real-time performance.

The mesh free model is worth mentioning. As the name may suggest, the model lacks an underlying mesh structure which defines the simulated body. The advantage of such a model is that it may support plasticity, fractures and splitting. There is no mesh which may be put in an awkward position so that it requires to be reconstructed. A nice example of a mesh free inspired model is [MKN+].

Sarah continues with a discussion of approximate continuum models which are more or less physics based and models with reduced degrees of freedom. There are several other models which fall in between the mentioned categories. Derivations of the mass-spring model and FEM model are probably the most common.

An interesting paper is the PhD thesis by Michael Hauth [Hau04]. It offers many information on cloth simulation, soft body simulation and related technical subjects as numerical time integration and continuum mechanics. The text is supported by many graphs and figures.

1.2 This paper

The work on my final thesis is a continuation of the final thesis of Maarten Everts [Eve06]. Under the guidance of Henk Bekker he developed a method which was based on finite differences and elementary continuum mechanics. It was named the CMPM which stands for Continuum Mechanics Particle Model. In this model point masses or particles lying centered in Voronoi cells interact with neighbouring particles based on principles from continuum mechanics. A contribution to this model described in this paper is the introduction of an implicit integrator. Also a lot of work was done on finding a way to make the Voronoi graph obsolete, however with limited success. During our work on the CMPM we found that the principles on which the CMPM calculates forces is based on false assumption. The model does not live up to the features it suggests, which raises questions on how useful the CMPM really is.

Some experiments were done to see if a simple form of volume preservation in a mass-spring model could lead to shearing forces in materials. The results of these experiments can be found in here.

Besides the efforts on the CMPM and volume preservation model this paper also goes into a different approach by treating the tetrahedra as a connective bond between the particles instead of springs. The advantage of moving from one dimensional springs to this type of three dimensional springs is that force calculations are based on full strain and stress tensors instead of much less versatile scalars. The virial theorem, applied in astrophysics and molecular dynamics amongst others, is normally used to find a pressure tensor based on the positions and forces of point masses, but also useful for our purposes. In the VIF (Virial Force) model, we first find the pressure tensor by the linear transformation that transforms the undeformed tetrahedron in the

deformed tetrahedron. Second we calculate deformation resisting forces for the vertices of the tetrahedron by the virial theorem.

After this introduction some basics of continuum mechanics are offered. Having a basic grasp of strains and stresses are necessary to understand the principles upon which the models in this paper are based. Next is a chapter which explains how these models are translated to the time discrete arena of the computer using numerical integration. In the next chapters our three models are discussed, starting with the inner workings of the CMPM, followed by the effects of simple volume preservation in a mass spring model and in the pre-final chapter we introduce the three dimensional spring models. In the concluding chapter we evaluate and give our opinion on the quality of all three models and recommendations for future research.

2 Continuum mechanics

Part of mathematical physics, the continuum mechanics branch is concerned with forces in a deformable body induced by external forces and the change of its shape. Originally brought to existence as a mathematical tool for correct structure analysis, the theories are now indespensable for bringing the real world into the computer.

Real world materials consist of a large number of interacting atoms. The forces between them, the way these molecules interact with each other defines the properties of the material. Continuum mechanics is an abstraction of reality and assumes that the matter is distributed continuously. Without this simplification calculation on material would be virtually impossible.

This chapter aims to aid in understanding the physical background of our physics based models. We focus on linear elastic materials, which are much easier to handle than non-linear elastic materials. No material is completely linear, but for simulations which are primarily intended for computer graphics applications, linear models mostly are sufficient.

We will take a quick look into the basic principles of continuum mechanics, the geometrical meaning of linear strain, and the derivation of the strain tensor. Next we introduce the stress tensor, how both tensors are related to each other and what role Young's modulus and Poisson's ratio play in this relation.

That we are not talking about matrices but about tensors is because there is a difference between the two. A tensor is a mathematical object which obeys certain transformation rules, whereas a matrix is a representation of a linear transformation. In this paper, this difference is irrelevant though.

2.1 Strain

The concept of strain is used to describe the distribution of deformation over a body. To explain what strain actually means, we will use a similar strategy as in [FB68].

We distinguish two different types of transformations on a body. There are transformations where the distance between points in the body are kept the same. Examples of these transformations are translation and rotation. Translation describes how points are moved from one place to the other without change of orientation. Rotation is about the orientation of a body. The other type of transformation is where the distance between two points does change. This transformation we call deformation and it describes how a volume changes shape. Deformation is the result of different displacements of elements within a single body.

2.1.1 Displacement

Consider a point at p, lying in a three dimensional body. Now this point is moved from its original position p to another position at q as in figure 2.1. This displacement is expressed by u, v and w and they correspond to the projection of the displacement in respectively the x, y and z axis. In Cartesian space the displacement can be calculated (2.1) by taking the difference between the new location and the original location.

$$\begin{aligned} u &= q_x - p_x \\ v &= q_y - p_y \\ w &= q_z - p_z \end{aligned} \tag{2.1}$$

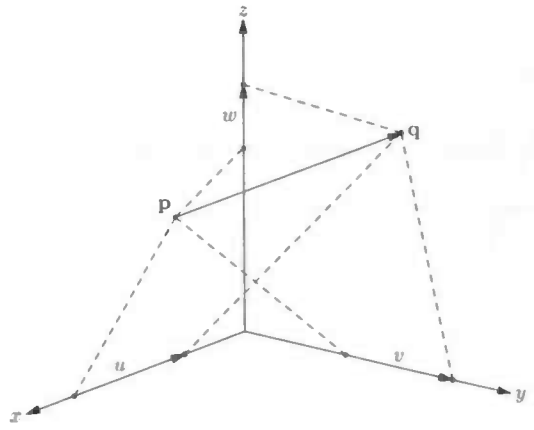


Figure 2.1: Displacement

2.1.2 Tensile deformation

Now we will zoom in on M and no longer consider it to be a point but rather a volume shaped as an infinitely small parallelepiped as in figure 2.2. This small body element will now not only be moved to another location by displacement, but also change shape as it is stretched and sheared. In this section we will first go into stretching or tensile deformation.

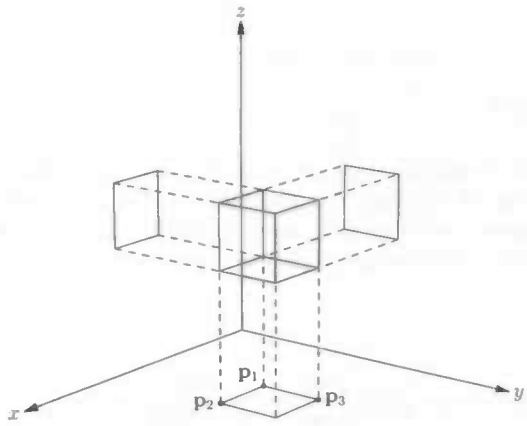


Figure 2.2: Parallelepiped

Consider the single edge p_1p_2 of this parallelepiped, parallel to the x-axis in its undeformed state. Figure 2.3 is a projection of edge p_1p_2 on the xy-plane together with its deformed equivalent q_1q_2. The displacement of point p_1 to q_1 along the x-axis is u (2.2). In case of the edge being undeformed the displacement p_2 would also move a distance u, yet here q_2 is displaced further, elongating the edge. This extra displacement of q_2 with respect to q_1 is expressed in terms of extra displacement u over a single unit on the x-axis multiplied by the length dx of edge p_1p_2 (2.3).

$$q_{1x} - p_{1x} = u \tag{2.2}$$

$$q_{2x} - p_{2x} = u + \frac{\partial u}{\partial x}dx \tag{2.3}$$

The difference in displacement is the tensile deformation, that in case of stretching will be positive and in case of compression negative. To describe the deformation of the material local to M in a general way, the length of

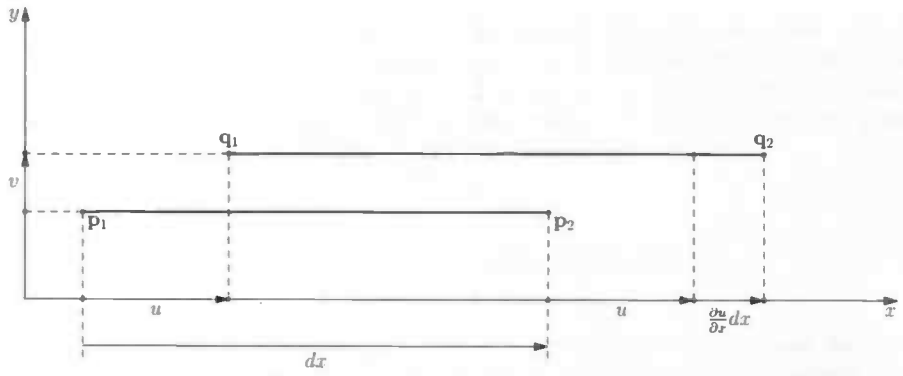


Figure 2.3: Tensile deformation

the edges are discarded from the equation (2.4) leaving only the unit elongations.

$$e_{xx} = \frac{\partial u}{\partial x} \tag{2.4}$$

In the same way the coefficients for tensile deformation can be found for the other axes (2.5).

$$\begin{aligned} e_{xx} &= \frac{\partial u}{\partial x} \\ e_{yy} &= \frac{\partial v}{\partial x} \\ e_{zz} &= \frac{\partial w}{\partial x} \end{aligned} \tag{2.5}$$

2.1.3 Angular deformation

In the previous section linear deformation was covered. This section is about angular deformation or shear strains. Again we will zoom in on M and look into the behaviour of single edges. Figure 2.4 shows edge $\mathbf{p}_1\mathbf{p}_2$ projected on the xy-plane combined with its deformed equivalent $\mathbf{q}_1\mathbf{q}_2$. In case of a small deformation, the angle α_{yx} can be approximated by the change of displacement in the y-axis over a single unit on the x-axis (2.6).

$$\alpha_{yx} \approx \tan \alpha_{yx} = \frac{\mathbf{q}_2\mathbf{q}_2{}'}{\mathbf{q}_1\mathbf{q}_2{}'} = \frac{\frac{\partial v}{\partial x} dx}{dx + \frac{\partial u}{\partial x} dx} = \frac{\frac{\partial v}{\partial x}}{1 + \frac{\partial u}{\partial x}} \tag{2.6}$$

Since we are talking small deformations, $\frac{\partial u}{\partial x}$ is so small compared to 1 that it can safely be discarded. This leaves only

$$\alpha_{yx} = \frac{\partial v}{\partial x}. \tag{2.7}$$

The second subscript of α denotes the axis of the edge in its original position and the first subscript is the axis towards the edge is rotating. The value of α_{yx} only partially describes the shear in the xy-plane. There is another shear angle; the edge $\mathbf{p}_1\mathbf{p}_3$ on the y-axis shearing towards the x-axis:

$$\alpha_{xy} = \frac{\partial u}{\partial y}. \tag{2.8}$$

The average of the two rotation angles together is the shear strain on the right angle $\mathbf{p}_1\mathbf{p}_2\mathbf{p}_3$ (2.10).

$$\varepsilon_{xy} = \frac{1}{2}(\alpha_{yx} + \alpha_{xy}) = \frac{1}{2}(\frac{\partial v}{\partial x} + \frac{\partial u}{\partial y}) \tag{2.9}$$

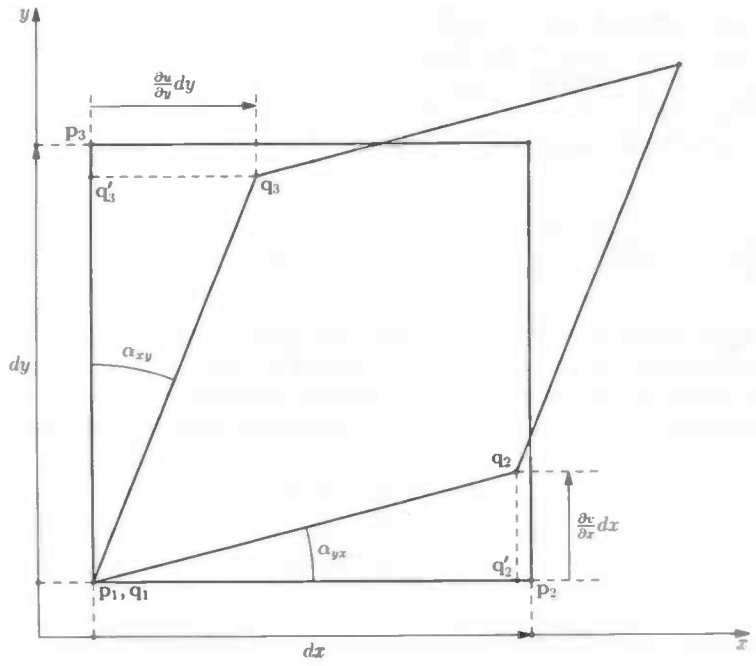
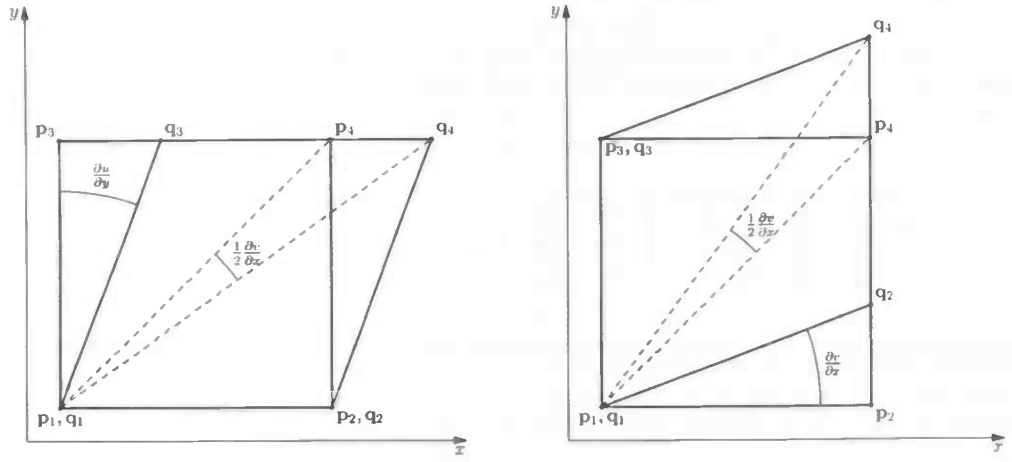


Figure 2.4: Angular deformation

Together with (2.5) the shear strains complete the six relations for deformation:

$$\begin{aligned} \varepsilon_{xy} &= \tfrac{1}{2}\left(\frac{\partial v}{\partial x} + \frac{\partial u}{\partial y}\right) \\ \varepsilon_{xz} &= \tfrac{1}{2}\left(\frac{\partial w}{\partial y} + \frac{\partial v}{\partial z}\right) \\ \varepsilon_{yz} &= \tfrac{1}{2}\left(\frac{\partial u}{\partial z} + \frac{\partial w}{\partial x}\right) \end{aligned} \tag{2.10}$$



(a) Rotation 1

(b) Rotation 2

Figure 2.5: Angular deformation

However, the story is not complete yet. When an angular deformation takes place, there is a small rotation taking place as well. When this is taken into account the strain becomes symmetrical. To demonstrate this, we look at the diagonal of a box undergoing an angular deformation. In figure 2.5 we see projections of two

differently sheared boxes, projected onto the xy-plane. The angular deformation causes the diagonal $\mathbf{q}_1\mathbf{q}_4$ of the sheared box to be tilted with respect to the diagonal $\mathbf{p}_1\mathbf{p}_4$. Let us assume that in the left figure, the angular deformation along the x-axis is α and the edges of the box are of size b. Then the angle between the two diagonals, the amount by which the box has rotated when counter clockwise rotations have positive angles, is approximately $-\frac{1}{2}\alpha$ for small deformations. This is confirmed in (2.11).

$$\frac{1}{4}\pi - \arctan\left(\frac{b + b\tan(\alpha)}{b}\right) = \frac{1}{4}\pi - \arctan(1 + \tan(\alpha)) \approx -\frac{1}{2}\alpha \tag{2.11}$$

For the figure on the right where shear takes place along the y-axis the result is almost the same, though the rotation is in the other direction. Under a deformation angle of α_{yx} rotation of the diagonal is $\frac{1}{2}\alpha_{yx}$. The sum of these two rotations, combined with the rotations by similar reasoning from the xz-plane and the yz-plane, one can find the total rotation ω around the three axes, resulting from angular deformation (2.12).

$$\begin{aligned} \omega_x &= \tfrac{1}{2}\left(\tfrac{\partial w}{\partial y} - \tfrac{\partial v}{\partial z}\right) \\ \omega_y &= \tfrac{1}{2}\left(\tfrac{\partial u}{\partial v} - \tfrac{\partial w}{\partial x}\right) \\ \omega_z &= \tfrac{1}{2}\left(\tfrac{\partial z}{\partial x} - \tfrac{\partial u}{\partial y}\right) \end{aligned} \tag{2.12}$$

Along the diagonal, the shape of the figure remains symmetrical during shear. Because of this, the strain tensor that is discussed in the following section, is also symmetrical.

2.1.4 Strain tensor

When a position vector local to the position the strain tensor was calculated for is multiplied by this strain tensor, the result is the amount of deformation that occurs at that point in all directions. The strain tensor is composed of the components found for tensile strain (2.5) and shear strain (2.10) and has the following form:

$$\varepsilon = \begin{pmatrix} \varepsilon_{xx} & \varepsilon_{xy} & \varepsilon_{xz} \\ \varepsilon_{xy} & \varepsilon_{yy} & \varepsilon_{yz} \\ \varepsilon_{xz} & \varepsilon_{yz} & \varepsilon_{zz} \end{pmatrix} \tag{2.13}$$

The nine partial derivatives in the tensor $\partial(u, v, w)/\partial(x, y, z)$ can be expressed as a combination of a symmetric and an asymmetric tensor. The symmetric tensor is the pure strain tensor whereas the asymmetric tensor contains the rotation.

$$\begin{pmatrix} \frac{\partial u}{\partial x} & \frac{\partial u}{\partial y} & \frac{\partial u}{\partial z} \\ \frac{\partial v}{\partial x} & \frac{\partial v}{\partial y} & \frac{\partial v}{\partial z} \\ \frac{\partial w}{\partial x} & \frac{\partial w}{\partial y} & \frac{\partial w}{\partial z} \end{pmatrix} = \begin{pmatrix} \varepsilon_{xx} & \varepsilon_{xy} & \varepsilon_{xz} \\ \varepsilon_{xy} & \varepsilon_{yy} & \varepsilon_{yz} \\ \varepsilon_{xz} & \varepsilon_{yz} & \varepsilon_{zz} \end{pmatrix} + \begin{pmatrix} 0 & -\omega_z & \omega_v \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{pmatrix} \tag{2.14}$$

Eleminating the asymmetric tensor which leaves the strain tensor is an important part of the models in chapter 6.

2.2 Stress

When external forces act upon a body this causes stress inside the body. Under stress a non-solid body may start to deform leading to strain. A relation between strain and stress exists and will be explained in the next section.

The formal definition of stress, which may also be referred to as pressure, is the amount of force per unit area (2.15). The amount of energy over a volume is equivalent to stress. Stress is defined for a point over an area,

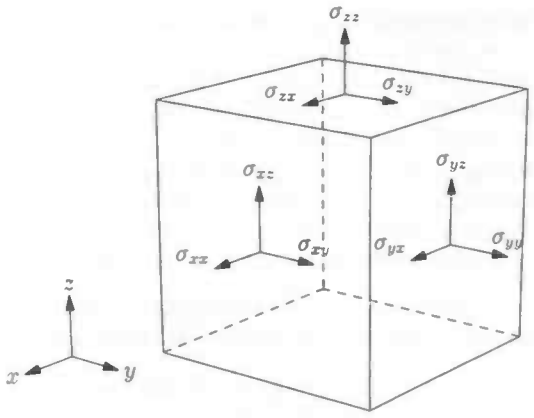


Figure 2.6: Stress tensor

facing in a certain direction. If we were to measure stress at a point inside a body into some direction, we would look at an infinitesimal cross section of the body, perpendicular to the concerned direction and through the point. The amount of force exerted perpendicular to the cross section determines the tensile stress and the amount of force parallel to the cross section the shear stress.

$$\frac{N}{m^2} = \frac{J}{m^3} \tag{2.15}$$

When the stress at a point is determined in the direction of all principal axes simultaneously we can place these results in a tensor. For three dimensions this stress tensor is a 3 × 3 matrix (2.16). This tensor has nine components but due to symmetry only six of them are unique. To build this tensor for some point, the stress is determined for each axis in the coordinate system by examining at that point the forces on a infinitesimal cross section perpendicular to the axis. For example, if we want to find the stress tensor for point p, we could cut out a infinitesimal cube aligned to the three coordinate axes (see figure 2.16). Next we measure the stress on the three faces whose normals point in the same direction as one of the positive axis. This way, one stress vector is found for each of the principal axes. The projections of these three stress vectors on the three coordinate axes together form the stress tensor. The subscript notation is explained as follows; the first subscript specifies the stress direction and the second subscript the axis on which this stress is projected.

$$\sigma = \begin{pmatrix} \sigma_{xx} & \sigma_{yx} & \sigma_{zx} \\ \sigma_{xy} & \sigma_{yy} & \sigma_{zy} \\ \sigma_{xz} & \sigma_{yz} & \sigma_{zz} \end{pmatrix} \tag{2.16}$$

As was mentioned before, the stress tensor is symmetrical like the strain tensor. This means that:

$$\begin{aligned} \sigma_{xy} &= \sigma_{yx} \\ \sigma_{xz} &= \sigma_{zx} \\ \sigma_{yz} &= \sigma_{zy}. \end{aligned} \tag{2.17}$$

An example of the usefulness of the stress tensor is that it can be used for calculating the force flux over a surface in any direction. Multiplying the stress tensor with a direction vector gives the force flux over a surface perpendicular to that direction. Multiply the stress with the area of the surface and one knows the force acting on that surface under the condition of the stress tensor.

2.3 Material parameters

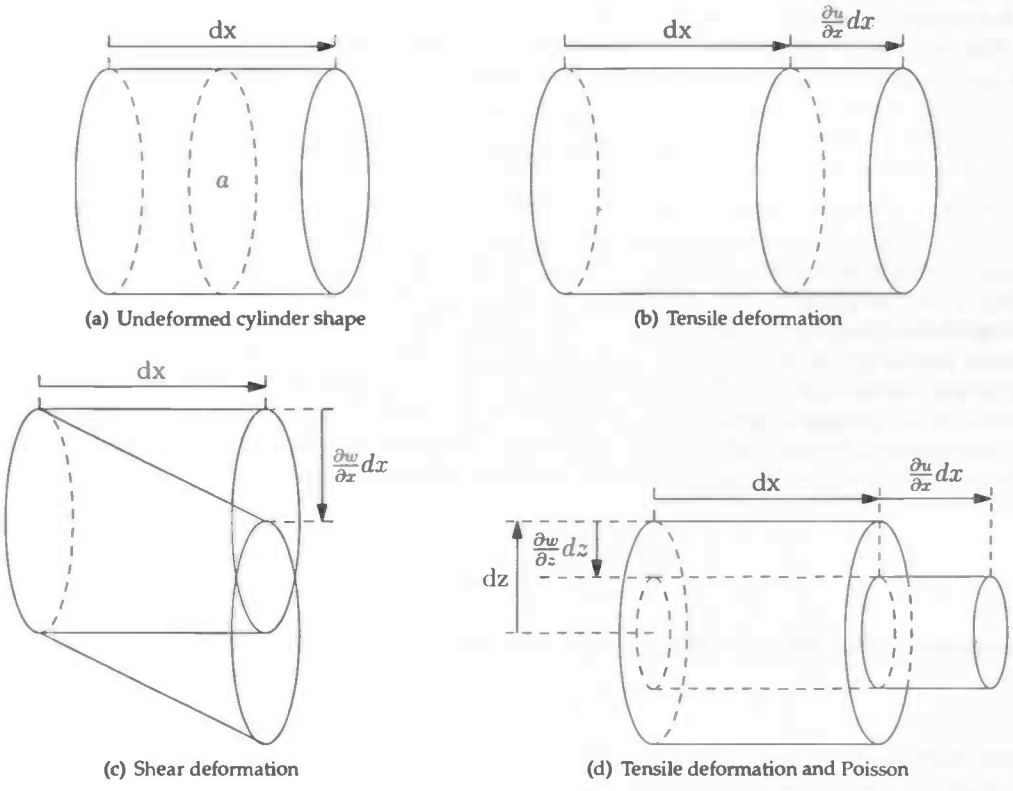
A material can have all kind of properties. In this paper only those that are concerned with the elastic behaviour of the material under stress are of interest. Furthermore, materials are considered to be linear and to have uniform properties over their entire volume. In other words, the material must be homogeneous.

The behaviour for an homogeneous isotropic material is defined by two constants, Young's modulus or the modulus of elasticity E and Poisson's ratio ν. Young's modulus is the rate by which a material resists tensile deformations and its unit is Pascal ($\frac{N}{m^2}$). The unit is equal to the unit for stress, which is not so strange when one considers the resistance to deformation as a counter-stress. For linear materials, Young's modulus remains constant under different amounts of stretching. Poisson's ratio describes the rate of expansion in each opposite direction when a material is compressed. For most materials Poisson's ratio ranges from 0 to 0.5. The ratio is a good measure for volume preservation, where a ratio of 0.5 indicates total volume preservation and a ratio of 0 no preservation at all. Though one might not expect a negative Poisson's ratio, such materials do exist. Examples of articles devoted to the negative Poisson's ratio are: [Eva88], [CE88b], [CE88a] and [Woj88].

Another material constant, the shear modulus G can be derived (2.18) from E and ν. The shear modulus is the rate by which a material resists to shear deformations and has the same unit as Young's modulus.

$$G = \frac{E}{2(1+\nu)} \tag{2.18}$$

We will show a simple demonstration of the workings of these material constants based on a beam shape, like a cylinder. Figure 2.7 shows a cylinder under different strain conditions.



(a) Undeformed cylinder shape

(b) Tensile deformation

(c) Shear deformation

(d) Tensile deformation and Poisson

Figure 2.7: An undeformed cylinder and several deformed cylinders.

The shape in resting position (figure 2.7(a)) has a length of dx and a cross section area of a. In figure 2.7(b) it is stretched so that its length increases with $\frac{\partial u}{\partial x}dx$. Young's modulus defines how strong the resisting force f_E in such a situation will be (2.19). This force is in the opposite direction as the stress that caused the deformation.

$$f_E = -\frac{\partial u}{\partial x}aE \tag{2.19}$$

When in figure 2.7(c) the shape is sheared instead of stretched, the shear modulus determines how strong the

force f_G is by which the material will try to return to its original state (2.20).

$$f_G = -\frac{\partial w}{\partial x} aG \tag{2.20}$$

When the material is compressed or stretched, Poisson's ratio determines how the material expands or contracts in transverse directions (2.21). Figure 2.7(d) shows an example of transverse contraction while the cylinder is being stretched.

$$\frac{\partial w}{\partial z} = -\nu \frac{\partial u}{\partial x} \tag{2.21}$$

These examples demonstrate how forces and stress are found from basic deformations using the material constants. For a strain tensor, where all kinds of deformations are interweaved, the relation of strain to stress is more complicated. The next section explains how stress can be found from the strain tensor.

2.4 Relation of strain to stress

Stress makes a material deform. A certain amount of deformation requires a certain amount of stress. Given some material properties it is possible to calculate to what extent a body will deform under specified stress conditions. It also works the other way around, one can calculate the amount of stress required to get a specified amount of deformation. This relation between strain and stress is sorted under the theory of elasticity. This section is about deriving a stress tensor when the strain tensor is known. Most information in this section was found in [Hei01].

Limiting ourselves to linear materials, Hooke's law states that for a linear material, the stress is proportional to the deformation. When this is generalized for all nine stress components and all nine strain components we find ourselves a $3 \times 3 \times 3 \times 3$ tensor. This fourth rank matrix describes the linear relationship between stress and strain (2.22). Such equations are also called constitutive equations.

$$\sigma_{ij} = c_{ijkl}\varepsilon_{kl} \tag{2.22}$$

There is also the reversed relation (2.23). In this paper we only have use for (2.22) to go from strain to stress, not the other way around. For that reason there is no further discussion on how stress relates to strain in here, but it is not very different from this discussion and can be found in [Hei01].

$$\varepsilon_{ij} = s_{ijkl}\sigma_{kl} \tag{2.23}$$

The matrix c with its 81 constants in it is not easy to work with, but luckily the symmetry of strain and stress reduce the complexity of the relation. When some assumptions are made about symmetry of the material, the matrix can be simplified even further. The largest reduction has to do with the symmetry of the strain and stress tensor. Because $\sigma_{ij} = \sigma_{ji}$ and $\varepsilon_{ij} = \varepsilon_{ji}$, we have

$$c_{ijkl} = c_{jikl} = c_{ijlk} = c_{jilk}. \tag{2.24}$$

The relation between the six unique components of the strain and stress tensor can now be described in only 36 entries instead of 81 (2.25).

$$\begin{pmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \sigma_{zz} \\ \sigma_{xy} \\ \sigma_{xz} \\ \sigma_{yz} \end{pmatrix} = \begin{pmatrix} c_{11} & c_{12} & c_{13} & c_{14} & c_{15} & c_{16} \\ c_{21} & c_{22} & c_{23} & c_{24} & c_{25} & c_{26} \\ c_{31} & c_{32} & c_{33} & c_{34} & c_{35} & c_{36} \\ c_{41} & c_{42} & c_{43} & c_{44} & c_{45} & c_{46} \\ c_{51} & c_{52} & c_{53} & c_{54} & c_{55} & c_{56} \\ c_{61} & c_{62} & c_{63} & c_{64} & c_{65} & c_{66} \end{pmatrix} \begin{pmatrix} \varepsilon_{xx} \\ \varepsilon_{yy} \\ \varepsilon_{zz} \\ \varepsilon_{xy} \\ \varepsilon_{xz} \\ \varepsilon_{yz} \end{pmatrix} \tag{2.25}$$

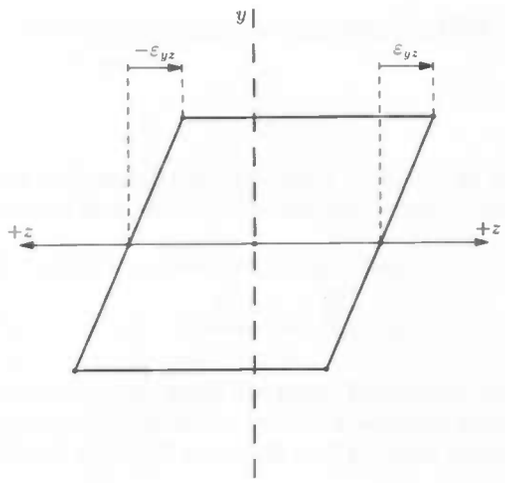


Figure 2.8: The value of ε_{yz} changes sign when the z-axis is reversed.

2.4.1 Planes of symmetry

A material can have one and two planes of symmetry. A material with one plane of symmetry is called an aeloptropic material and a material with two orthogonal planes of symmetry an orthotropic material. From having two planes of symmetry follow automatically three planes of symmetry, where the third plane is the consequence of a combination of the symmetry in the other two planes.

For a material that has a plane of symmetry on the xy-plane, the direction of the z-axis can safely be reversed without invalidating equation (2.25). When the strain or stress is examined along the negative z-axis, we see that the shear components for strain $\{\varepsilon_{xz}, \varepsilon_{yz}\}$ and stress $\{\sigma_{xz}, \sigma_{yz}\}$ change sign (figure 2.8). The components $\{\varepsilon_{xx}, \varepsilon_{yy}, \varepsilon_{xy}\}$ and $\{\sigma_{xx}, \sigma_{yy}, \sigma_{xy}\}$ are unchanged and so are ε_{zz} and σ_{zz}, because tensile strain and stress is opposite in both directions along the z-axis. That changing the sign of $\{\varepsilon_{xz}, \varepsilon_{yz}\}$ always results in the change of sign of only $\{\sigma_{xz}, \sigma_{yz}\}$ and no other elements of σ has some implications for the elements in (2.25). First of all it means that $\{\sigma_{xz}, \sigma_{yz}\}$ are only depending on $\{\varepsilon_{xz}, \varepsilon_{yz}\}$. Second it means that the other elements of σ, $\{\sigma_{xx}, \sigma_{yy}, \sigma_{zz}, \sigma_{xy}\}$ are not depending on $\{\varepsilon_{xz}, \varepsilon_{yz}\}$ at all. This is the reason why many elements of (2.25) are zero (2.26) for an aeloptropic material.

$$\begin{pmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \sigma_{zz} \\ \sigma_{xy} \\ \sigma_{xz} \\ \sigma_{yz} \end{pmatrix} = \begin{pmatrix} c_{11} & c_{12} & c_{13} & c_{14} & 0 & 0 \\ c_{21} & c_{22} & c_{23} & c_{24} & 0 & 0 \\ c_{31} & c_{32} & c_{33} & c_{34} & 0 & 0 \\ c_{41} & c_{42} & c_{43} & c_{44} & 0 & 0 \\ 0 & 0 & 0 & 0 & c_{55} & c_{56} \\ 0 & 0 & 0 & 0 & c_{65} & c_{66} \end{pmatrix} \begin{pmatrix} \varepsilon_{xx} \\ \varepsilon_{yy} \\ \varepsilon_{zz} \\ \varepsilon_{xy} \\ \varepsilon_{xz} \\ \varepsilon_{yz} \end{pmatrix} \tag{2.26}$$

For an orthotropic material there is a second plane of symmetry, say the yz-plane. By similar reasoning as for the symmetry for the xy-plane, we see that when $\{\varepsilon_{xy}, \varepsilon_{xz}\}$ change sign, on the left side only $\{\sigma_{xy}, \sigma_{xz}\}$ change sign. All the other elements of σ remain unchanged. Combined with (2.26), we can say that σ_{xy} only depends on ε_{xy}, σ_{xy} only depends on ε_{xy} and σ_{yz} only depends on ε_{yz}. Also, $\{\sigma_{xx}, \sigma_{yy}\sigma_{zz}\}$ are not depending on $\{\varepsilon_{xy}, \varepsilon_{xz}, \varepsilon_{yz}\}$. When the corresponding elements in (2.26) are set to zero (2.27), only 12 non-zero constants remain.

$$\begin{pmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \sigma_{zz} \\ \sigma_{xy} \\ \sigma_{xz} \\ \sigma_{yz} \end{pmatrix} = \begin{pmatrix} c_{11} & c_{12} & c_{13} & 0 & 0 & 0 \\ c_{21} & c_{22} & c_{23} & 0 & 0 & 0 \\ c_{31} & c_{32} & c_{33} & 0 & 0 & 0 \\ 0 & 0 & 0 & c_{44} & 0 & 0 \\ 0 & 0 & 0 & 0 & c_{55} & 0 \\ 0 & 0 & 0 & 0 & 0 & c_{66} \end{pmatrix} \begin{pmatrix} \varepsilon_{xx} \\ \varepsilon_{yy} \\ \varepsilon_{zz} \\ \varepsilon_{xy} \\ \varepsilon_{xz} \\ \varepsilon_{yz} \end{pmatrix} \tag{2.27}$$

2.4.2 Axes of symmetry

Besides planes of symmetry there can also be axes of symmetry. A material with one axis of symmetry is called an hexagonal material. When there is also a second axis of symmetry the third axis automatically becomes symmetric as well. This results in a material that is fully symmetric and is referred as an isotropic material.

With one axis of symmetry, the material can be rotated around that axis, without invalidating the equation (2.27). Take the x-axis, around which the body is rotated 90 degrees clockwise. In case the material is symmetric in that x-axis, the following must still be true:

$$\begin{pmatrix} \sigma_{xx} \\ \sigma_{zz} \\ \sigma_{yy} \\ \sigma_{xz} \\ -\sigma_{xy} \\ -\sigma_{yz} \end{pmatrix} = \begin{pmatrix} c_{11} & c_{12} & c_{13} & 0 & 0 & 0 \\ c_{21} & c_{22} & c_{23} & 0 & 0 & 0 \\ c_{31} & c_{32} & c_{33} & 0 & 0 & 0 \\ 0 & 0 & 0 & c_{44} & 0 & 0 \\ 0 & 0 & 0 & 0 & c_{55} & 0 \\ 0 & 0 & 0 & 0 & 0 & c_{66} \end{pmatrix} \begin{pmatrix} \varepsilon_{xx} \\ \varepsilon_{zz} \\ \varepsilon_{yy} \\ \varepsilon_{xz} \\ -\varepsilon_{xy} \\ -\varepsilon_{yz} \end{pmatrix} \tag{2.28}$$

In (2.28) we can see some possibilities for simplification of the matrix c. First of all, when the two sets of elements $\{\varepsilon_{xy}, \varepsilon_{xz}\}$ swap rows, the outcome for $\{\sigma_{xy}, \sigma_{xz}\}$ does not change. Therefore we must conclude that $c_{55} = c_{44}$. Also, the elements $\{\varepsilon_{yy}, \varepsilon_{zz}\}$ can swap rows without affecting the value for the elements $\{\sigma_{xx}, \sigma_{yy}, \sigma_{zz}\}$. From the fact that σ_{xx} stays the same we can conclude that $c_{13} = c_{12}$. We also see that the two sets of equations for both σ_{yy} and σ_{zz}

$$\sigma_{yy} = c_{21}\varepsilon_{xx} + c_{22}\varepsilon_{yy} + c_{23}\varepsilon_{zz} \tag{2.29}$$
$$\sigma_{yy} = c_{31}\varepsilon_{xx} + c_{33}\varepsilon_{yy} + c_{32}\varepsilon_{zz} \tag{2.30}$$
$$\sigma_{zz} = c_{31}\varepsilon_{xx} + c_{32}\varepsilon_{yy} + c_{33}\varepsilon_{zz} \tag{2.31}$$
$$\sigma_{zz} = c_{21}\varepsilon_{xx} + c_{23}\varepsilon_{yy} + c_{22}\varepsilon_{zz} \tag{2.32}$$

must hold. This is only accomplished when $c_{31} = c_{21}$, $c_{33} = c_{22}$ and $c_{32} = c_{23}$, leading us to the reduced constitutive equation

$$\begin{pmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \sigma_{zz} \\ \sigma_{xy} \\ \sigma_{xz} \\ \sigma_{yz} \end{pmatrix} = \begin{pmatrix} c_{11} & c_{12} & c_{12} & 0 & 0 & 0 \\ c_{21} & c_{22} & c_{23} & 0 & 0 & 0 \\ c_{21} & c_{23} & c_{22} & 0 & 0 & 0 \\ 0 & 0 & 0 & c_{44} & 0 & 0 \\ 0 & 0 & 0 & 0 & c_{44} & 0 \\ 0 & 0 & 0 & 0 & 0 & c_{66} \end{pmatrix} \begin{pmatrix} \varepsilon_{xx} \\ \varepsilon_{yy} \\ \varepsilon_{zz} \\ \varepsilon_{xy} \\ \varepsilon_{xz} \\ \varepsilon_{yz} \end{pmatrix}. \tag{2.33}$$

Another axis of symmetry makes the material completely symmetric. By similar reasoning as above the equation for a completely symmetric or isotropic material (2.34) only relies on three different constants $\{c_{11}, c_{12}, c_{44}\}$.

$$\begin{pmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \sigma_{zz} \\ \sigma_{xy} \\ \sigma_{xz} \\ \sigma_{yz} \end{pmatrix} = \begin{pmatrix} c_{11} & c_{12} & c_{12} & 0 & 0 & 0 \\ c_{12} & c_{11} & c_{12} & 0 & 0 & 0 \\ c_{12} & c_{12} & c_{11} & 0 & 0 & 0 \\ 0 & 0 & 0 & c_{44} & 0 & 0 \\ 0 & 0 & 0 & 0 & c_{44} & 0 \\ 0 & 0 & 0 & 0 & 0 & c_{44} \end{pmatrix} \begin{pmatrix} \varepsilon_{xx} \\ \varepsilon_{yy} \\ \varepsilon_{zz} \\ \varepsilon_{xy} \\ \varepsilon_{xz} \\ \varepsilon_{yz} \end{pmatrix}. \tag{2.34}$$

2.4.3 Elastic constants based on material parameters

The previous sections have explained how a linear relation between strain and stress based on 81 constants 2.22 can be reduced to a relation of only 3 constants (2.34) when the material is isotropic. In [Hei01] the values

for these three constants c_{ij} are derived and based on Young's modulus E and Poisson's ratio ν. So at this point, E and ν are the only free parameters left.

$$\begin{aligned} c_{11} &= \frac{E(1-\nu)}{(1+\nu)(1-2\nu)} \\ c_{12} &= \frac{E\nu}{(1+\nu)(1-2\nu)} \\ c_{44} &= \frac{E}{1+\nu} \end{aligned} \tag{2.35}$$

When the constitutive equation is written out fully for all the values of σ we get:

$$\begin{aligned} \sigma_{xx} &= \frac{E}{(1+\nu)(1-2\nu)}[(1-\nu)\varepsilon_{xx} + \nu(\varepsilon_{yy} + \varepsilon_{zz})] \\ \sigma_{yy} &= \frac{E}{(1+\nu)(1-2\nu)}[(1-\nu)\varepsilon_{yy} + \nu(\varepsilon_{zz} + \varepsilon_{xx})] \\ \sigma_{zz} &= \frac{E}{(1+\nu)(1-2\nu)}[(1-\nu)\varepsilon_{zz} + \nu(\varepsilon_{yy} + \varepsilon_{xx})] \\ \sigma_{xy} &= \frac{E}{1+\nu}\varepsilon_{xy} \\ \sigma_{xz} &= \frac{E}{1+\nu}\varepsilon_{xz} \\ \sigma_{yz} &= \frac{E}{1+\nu}\varepsilon_{yz} \end{aligned} \tag{2.36}$$

2.5 Summary

In this chapter some continuum mechanics principles were introduced. First there is strain, normally expressed in the form of a tensor which describes the deformation at a point inside a body. We have covered how the strain tensor is composed from tensile and angular strain components. Second there is stress, which is also defined for a point in the form of a tensor and describes the force that causes deformation. Then we introduced Young's modulus, Poisson's ratio and the shear modulus and how these are related to each other. Finally we explained how the strain tensor is related to the stress tensor for isotropic materials, based on material parameters.

3 Computational physics

The field of computational physics is concerned with simulating processes found in physics on computers. This chapter is not a full course on all types of problems involved with computational physics, but only those that matter to our models in the following chapters. What will be discussed in here is how to numerically integrate interacting point masses over time.

In the computer models found in the following chapters, the volume is discretized in point masses. In computer simulations it is very common to make use of points masses, where a volume's mass is concentrated in a local point called center of mass or center of gravity. In real life this can not happen because assigning any mass to something that has no volume would suggest an infinite density. What makes it so commonly used is that point masses are much easier to handle than volumes. Justification lies in the fact that a (approximately) spherical body behaves to certain extent similar to the situation where all its mass is packed at its center point. In this paper point masses may also be referred to as particles.

For most particle based dynamic systems with more than two particles, the future state of a system can not be calculated based on its current state with analytical methods. What can be done is to establish how the system is changing at the current state in the form of differential equations. The partial differential equations we need were covered in the previous chapter about continuum mechanics. Numerical integration is about how we move in time from one state of a dynamical system to an approximation of a future state.

In the first sections important concepts of numerical integration are introduced. After the discussion of numerical integration in general, we move on to where integration is involved in finding velocities and positions of particles. After particle forces are found, basic Newtonian Mechanics are the foundations by which the positions of the particles are updated every time step. To have some sort of cohesion between these particles there must be some interaction between them. To demonstrate how to combine these concepts into a working simulation we will illustrate this by using a mass-spring example. There will be a full explanation of an explicit integration approximation to the solution and an introduction to the implicit integration solution as it was proposed by David Baraff and Andrew Witkin in [BW98].

3.1 Introduction numerical integration

The state of a deformable body in most real world simulations is described by the position and velocity of its matter. Forces are calculated as a function of these positions and velocities. Over time these forces change the velocity which over time changes the position points inside the material. This system of differential equations needs to be solved for a given starting position. In this case, where there are so many interactions, an analytical solution can quickly be ruled out. A decent numerical approximation is what should be looked for.

The reason why we are talking about numerical approximation and not about numerical solution is because the results from numerical methods may diverge from the exact solution. Computers have limited resolution for storing real numbers and errors in intermediate results may lead to even larger errors in subsequent results. To keep the error under control, one must make decisions in what methods to use and how to set the parameters involved with the numerical calculations. The right balance between accuracy and computing time must be found.

Also, numerical approximations do not provide us with continues results. Instead, approximations are calculated for discrete moments in time only. Both volume and time must be discretized.

For initial value problems, there are a number of integration methods available. One common categorization of integration methods are the explicit and implicit methods. Important when making a decision which category of methods is best suited is to consider whether or not the differential equation at hand are 'stiff' equations. This will be explained later.

3.2 First order Euler integration

In this section we will discuss the first order Euler integration methods. This is the simplest tool for numerical integration available, but also provides poor accuracy in comparison to more advanced methods. The simplicity however makes it very easy to handle and understand. First we will derive two first order Euler equations and then discuss some of their properties. Most of this knowledge was found in [BF05].

To derive the Euler integration method the Taylor series are used. All we have is a differential equation of a function and the function itself is what we want to know. A function about a point t_0 can be expressed in the form of a Taylor series (3.1).

$$y(t) = y(t_0) + (t - t_0)\dot{y}(t_0) + \frac{(t - t_0)^2}{2!}\ddot{y}(t_0) + ... + \frac{(t - t_0)^n}{n!}y^{(n)}(t_0), n = 1, 2, 3, ... \tag{3.1}$$

The higher order terms are less significant then the lower order terms. Using this argument we discard all the terms from second order and beyond. This will only leave the function which is based on its first derivative $\dot{y}$ and the value of y at position t_0 Discarding the higher order terms leads to truncation error when the difference between t and t_0 is not infinitesimal. The larger the distance between t and t_0 is, the larger the truncation error will be. The solution for $y(t_1)$ one step h beyond $y(t_0)$ becomes:

$$y(t_1) = y(t_0 + h) \approx y(t_0) + h\dot{y}(t_0). \tag{3.2}$$

This equation is the basis for the explicit Euler method. This method may also be referred to as the forward Euler method, since it is based on forward differences $t_{i+1} = t_i + h$. Methods are called explicit when the next result $y(t_{i+1})$ is based entirely on value at t_i or prior to t_i, in contrast to implicit methods. The equation for the backward or implicit Euler method can be found by using backward differences: $t_i = t_{i+1} - h$. With the Taylor series (3.1) we use $y(t_1)$ to find an approximation (3.3) for $y(t_0)$.

$$y(t_0) = y(t_1 - h) \approx y(t_1) - h\dot{y}(t_1) \tag{3.3}$$

But in this case we do not want to go back in time so we solve (3.3) for $y(t_1)$ (3.4). The forward and backward equations may look very similar but the small difference between them has some substantial implications.

$$y(t_1) = y(t_0 + h) \approx y(t_0) + h\dot{y}(t_1) \tag{3.4}$$

With a starting value for $y(t_0)$ and using either equation 3.2 or 3.4 one can calculate subsequent results $y(t_{i+1})$ by using $y(t_i)$. Making h infinitesimal will still result in a continues and exact solution. This is not practical for computers so we need to discretize. Instead of using infinitesimal differences h, we will start using finite differences. This is where the truncation error kicks in and the result starts being an approximation instead of a solution. Subsequent results may gradually drift from the exact solution. Because the first order term of the Taylor series remained, Euler integration is known to be accurate until the first order of differentiation and is therefore called a first order integration method.

To demonstrate the difference between the numerical approximations of both the explicit and implicit Euler methods and the exact analytical solution they are compared for a test function $\dot{y} = \lambda y$ with $\lambda < 0$. The begin value $y(t_0) = \alpha$ is set to 1. The exact solution of y is $y = \alpha e^{\lambda x}$. The results are plotted in figure 3.1 with the exact solution represented by a dotted line.

What can be seen here is that the explicit Euler method overshoots the curve of the solution whereas the implicit Euler method undershoots the curve. Taking a smaller time step will make this effect less pronounced. How this behaviour might affect the outcome over a longer period can be demonstrated with the integration of a wave function. Figure 3.2 shows how the wave equation behaves for implicit and explicit Euler integration.

The explicit method gains amplitude over time and the implicit method fades out. Both methods are not a good choices for use on a wave equation but the internal damping the implicit method demonstrates here makes that it is more stable than the explicit method.

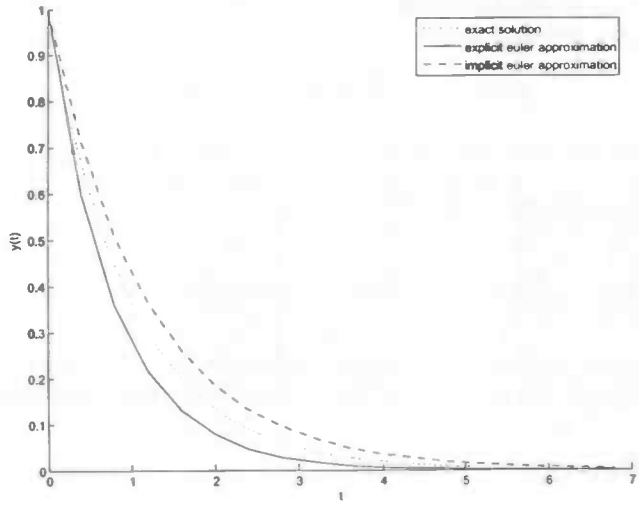
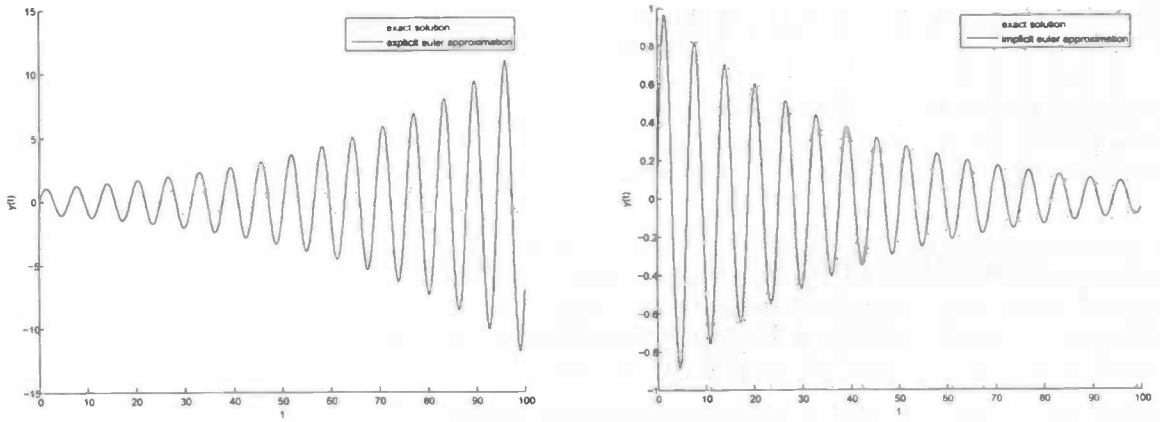


Figure 3.1: Explicit and implicit Euler approximation of the test function



(a) Explicit Euler

(b) Implicit Euler

Figure 3.2: Wave function approximated by different numerical integration methods.

3.3 Stiff equations

As mentioned before, it is important when integrating numerically to know if differential equations are 'stiff'. This gives insight how stable the results of a specific numerical method will be, in the sense that small perturbations produce correspondingly small changes in subsequent approximations. A numerical integration method that is incapable of handling stiff equations is likely to 'explode' with an uncontrolled error dominating the results.

This stiffness is a relative term. Stiff differential equations have an exact solution that contains a term in the form of $e^{-\lambda x}$ where λ is relatively large. The n-th derivative of such a function is $\lambda^n e^{-\lambda x}$ where the additional factor λ^n reduces the decay. The size of the time step h determines how high the influence is that the higher derivatives have on the approximation. When the time step is taken too large, the error term which builds up during the integration process is multiplied by the large factor at every step and may cause the approximation to increase very fast.

To see how capable an integration method is in handling stiff equations, we can test it by looking at the test

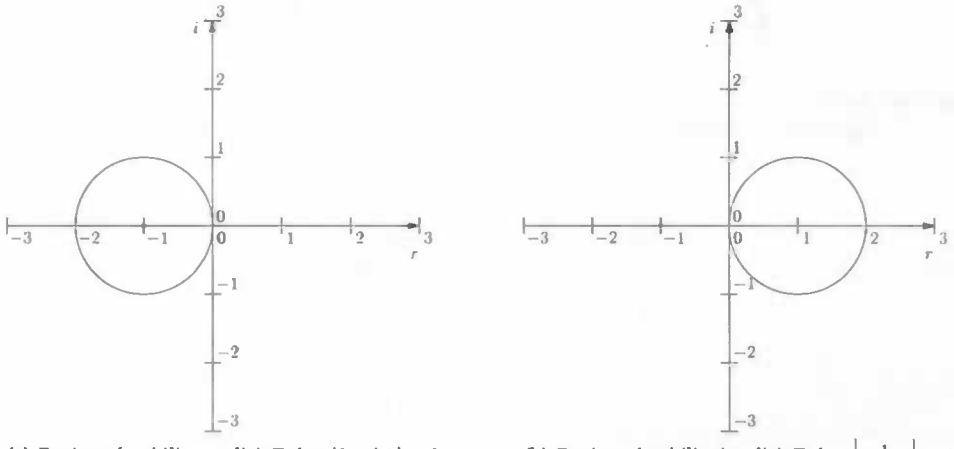
equation $\dot{y} = \lambda y$ with $\lambda < 0$ again. In (3.5) the explicit Euler formula for the approximation of this equation is given. The size of the time step is h and the i-th approximation is denoted by w_i.

$$w_{i+1} = w_i + h\lambda w_i = (1 + h\lambda)w_i \tag{3.5}$$

Because $\lambda < 0$ we know that the exact solution decreases over time. In (3.5) the absolute value for w_{i+1} will only be smaller than w_i if $|1 + h\lambda| < 1$. Figure 3.3(a) shows which values in the complex plane for $h\lambda$ are safe, which is called the region of stability.

$$w_{i+1} = w_i + h\lambda w_i = \left(\frac{1}{1 - h\lambda}\right) w_i \tag{3.6}$$

For the implicit Euler can we do the same. The implicit Euler formula for the test equation (3.6) shows that for stability it is required that $|\frac{1}{1-h\lambda}| < 1$. Figure 3.3(b) shows graphically what form this region has.



(a) Region of stability explicit Euler: $|1 + h\lambda| < 1$

(b) Region of stability implicit Euler: $\left|\frac{1}{1-h\lambda}\right| < 1$

Figure 3.3: The region of stability in these graphs is colored.

Though it is not always this easy to find the exact region of stability for integration methods, sometimes a good indication is all one needs to make an educated decision which integration method to use. For the implicit Euler integrator one can see that its region of stability is much larger than it is for explicit Euler integrator. Integration methods, like implicit Euler, for which the entire left side of the complex plane fall into the region of stability are labeled A-stable. These class methods are very capable of coping with stiff equations.

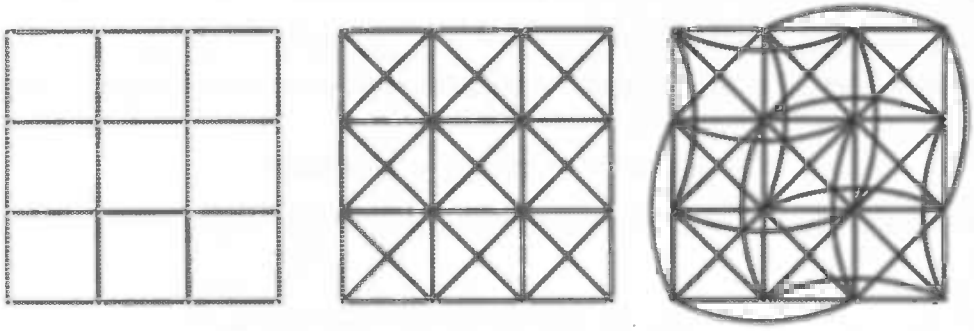
3.4 Introduction Mass-spring model

For clarification of the discussion about integrating point masses, we rely on the demonstration of a 3-dimensional mass-spring model. The mass-spring model lends itself for simple rope, cloth and volume simulations. The basic idea is that a set of n particles are connected to each other by a set of springs, by which the particles can interact. Each spring tries to keep the particles at equal distance from each other. When two connected particles are moved in opposite directions, the spring will respond by a counteracting force. Depending on the direction of the forces, the spring will push the particles away or pull the particles together in an attempt to maintain the distance there originally was between the particles.

An example of forces acting from outside the body on the particles inside is gravity (3.7). The vector gravity constant g times the mass m of the particle gives the force $\mathbf{f}_g$ resulting from gravity. Other examples are forces originating from collisions and magnetic fields amongst others.

$$\mathbf{f}_g = m\mathbf{g} \tag{3.7}$$

Though all kinds of connection schemes are possible, what is seen mostly is that particles are connected to spatially local particles only. One reason not to connect to any more particles than the neighbouring ones is that the computational cost of the simulation depends heavily on the number of springs. Interactions over a longer distance take place by indirect contact through multiple springs. See figure 3.4 for examples of connection schemes.



(a) Tensile springs (b) Shear springs (c) Bend springs

Figure 3.4: Two dimensional connection schemes as they are used for many mass-spring based cloth simulations. Tensile springs 3.4(a) resist stretching. Additional shear springs 3.4(b) may prevent the cloth from collapsing on itself. Bend springs 3.4(c) resist folding.

Each particles has a certain mass. When each particle represents the mass of a piece of material, a common strategy to distribute the mass over the particles is to assign the mass of each element to the closest particle. This is for example one of the applications of the Voronoi graph in the CMPM, which can be read about in the following chapter.

A spring is placed between two particles. The parameters of the spring are the spring constant k and the initial length l of the spring. The spring constant k determines the amount of counteracting force by which the spring responds to changes in length. The original length must be known to response to any change in length. Consider $\mathbf{p}$ and $\mathbf{q}$ as the positions of two spring-connected particles and $\hat{\mathbf{n}}$ is the unit direction of the spring from particle $\mathbf{p}$ to $\mathbf{q}$. Based on Hooke's law for springs, the formulas for the forces $\mathbf{f}_p$ and $\mathbf{f}_q$ acting on both particles caused by the spring then becomes:

$$\begin{aligned} \mathbf{f}_p &= k\hat{\mathbf{n}}(|\mathbf{q}-\mathbf{p}|-l) \\ \mathbf{f}_q &= -k\hat{\mathbf{n}}(|\mathbf{q}-\mathbf{p}|-l) \end{aligned} \tag{3.8}$$

The model can be advanced by adding an additional damping force to the spring. We define the damping force as a resisting force to difference in velocity $\mathbf{v}$ between a set of interacting particles by some global constant b (3.9). This spring damping force acts only on velocities in the same direction of the spring. In real life damping will occur due to friction where kinetic energy is lost to heat. With some integration methods, like the explicit Euler method, the presence of damping may be necessary for the model to remain stable.

$$\begin{aligned} \mathbf{f}_p &= b(\hat{\mathbf{n}}\cdot(\mathbf{v}_q-\mathbf{v}_p))\hat{\mathbf{n}} \\ \mathbf{f}_q &= -b(\hat{\mathbf{n}}\cdot(\mathbf{v}_q-\mathbf{v}_p))\hat{\mathbf{n}} \end{aligned} \tag{3.9}$$

The total force on each particle is the sum of all forces from connected springs combined with spring damping forces and external forces like those originating from collisions of particles with foreign objects and gravity.

3.5 Mass-spring model numerical integration

In here we will continue with a discussion on the integration of forces and velocities over time. Both explicit and implicit methods as described in section 3.2 are covered.

To keep notation consistent, a similar form as found in [BW98] is used throughout this section. Bold lower case math symbols are vectors and bold upper case math symbols are matrices. Bold-faced subscript are time indices, otherwise they are vector indices. The particle position vector is denoted as $\mathbf{x}$, the particle velocity vector as $\mathbf{v}$ and the particle force vector as $\mathbf{f}$. The positions, velocities and forces for all particles are combined in a single large $3n$ vector as shown in (3.10) for $\mathbf{x}$, where n is the total number of particles and x_i is the position of the i-th particle.

$$\mathbf{x} = (x_{1x}, x_{1y}, x_{1z}, x_{2x}, x_{2y}, x_{2z}, ..., x_{nx}, x_{ny}, x_{nz})^{\mathrm{T}} \tag{3.10}$$

The mass of all the particles is denoted as the $3n \times 3n$ matrix M. Only the diagonal of M is non-zero and is filled as in (3.11), where m_i is the mass for the i-th particle.

$$\mathrm{M} = \mathrm{diag}(m_1, m_1, m_1, m_2, m_2, m_2, ...m_n, m_n, m_n) \tag{3.11}$$

Most of the time we will need the inverse mass matrix M^{-1} which is, since it is a diagonal matrix, not hard to find (3.12):

$$\mathrm{M}^{-1} = \mathrm{diag}\left(\frac{1}{m_1}, \frac{1}{m_1}, \frac{1}{m_1}, \frac{1}{m_2}, \frac{1}{m_2}, \frac{1}{m_2}, ...\frac{1}{m_n}, \frac{1}{m_n}, \frac{1}{m_n}\right). \tag{3.12}$$

In each of the following subsections the application of one integration method on the mass-spring model will be explained.

3.5.1 Explicit Euler integration

In Newtonian Physics, the system of derivatives by which particle velocity and position are determined can be expressed as:

$$\frac{d}{dt}\begin{pmatrix}\mathbf{x}\\ \mathbf{v}\end{pmatrix} = \begin{pmatrix}\mathbf{v}\\ \mathrm{M}^{-1}\mathbf{f}(\mathbf{x},\mathbf{v})\end{pmatrix}. \tag{3.13}$$

The force vector $\mathbf{f}$ is a function of the particle's positions $\mathbf{x}$ and velocities $\mathbf{v}$. The last three chapters of this paper will be about different implementations for this function $\mathbf{f}(\mathbf{x}, \mathbf{v})$. In this section we will simply use the in section 3.4 described forces for the mass-spring model. A time discretized explicit Euler approximation of (3.13) is not too hard to find. When $\Delta\mathbf{x}$ and $\Delta\mathbf{v}$ are respectively the approximations for change in position and velocity over a time h, the position and velocity in the next time step are:

$$\begin{aligned}\mathbf{x_1} &= \mathbf{x_0} + \Delta\mathbf{x}\\ \mathbf{v_1} &= \mathbf{v_0} + \Delta\mathbf{v}\end{aligned} \tag{3.14}$$

With the time derivatives in (3.13), we can find the values for $\Delta\mathbf{x}$ and $\Delta\mathbf{v}$:

$$\begin{pmatrix}\Delta\mathbf{x}\\ \Delta\mathbf{v}\end{pmatrix} = h\begin{pmatrix}\mathbf{v_0}\\ \mathrm{M}^{-1}\mathbf{f}(\mathbf{x_0},\mathbf{v_0})\end{pmatrix}. \tag{3.15}$$

When the equations (3.14) and (3.15) are combined, equation (3.2) for calculating the values for position and velocity in the next time step emerge. Using explicit Euler integration we get

$$\begin{pmatrix}\mathbf{x_1}\\ \mathbf{v_1}\end{pmatrix} = \begin{pmatrix}\mathbf{x_0}\\ \mathbf{v_0}\end{pmatrix} + h\begin{pmatrix}\mathbf{v_0}\\ \mathrm{M}^{-1}\mathbf{f}(\mathbf{x_0},\mathbf{v_0})\end{pmatrix}. \tag{3.16}$$

Now we have all we need to develop an algorithm that calculates the positions of the particles for consecutive time steps. First we need to set the initial state of the mass-spring model, so that all parameters at simulation time t_0 are known. To calculate the state of the model after taking one time step h we take the following actions:

- Calculate the forces $\mathbf{f}(\mathbf{x_i}, \mathbf{v_i})$ acting on the particles.
- Update the position: $\mathbf{x_{i+1}} = \mathbf{x_i} + h\mathbf{v_i}$.
- Update the velocity: $\mathbf{v_{i+1}} = \mathbf{v_i} + h\mathrm{M}^{-1}\mathbf{f}(\mathbf{x_i}, \mathbf{v_i})$.

After updating the position and velocity the whole process can be repeated for the next time step.

3.5.2 Leapfrog integration

For slightly better results we could use the leapfrog integration method virtually without extra costs. Leapfrog integration is an integration method popular in the computer physics field. The cost is virtually the same as for explicit Euler integration, but the approximation results are slightly better. Another interesting fact is that it leapfrog is time reversible. By taking steps backward in time one can retrieve earlier simulation results. Since this method will be mentioned later on, it should also be mentioned here. We also introduce a little variant on leapfrog integration which we named 'lazy leapfrog'. This method earns its 'lazy' additive from the fact that one can go from the explicit Euler integration to lazy leapfrog integration by simply switching two blocks of code.

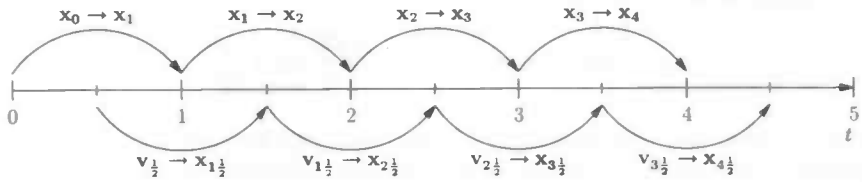


Figure 3.5: Leapfrog integration

With the original leapfrog method, the velocity is no longer calculated for the same time steps as the positions are. It is calculated for the time in the middle of the interval between two consecutive position calculations. The next position value $\mathbf{x_1}$ is calculated with $\mathbf{v}_{\frac{1}{2}}$, then the next velocity $\mathbf{v}_{1\frac{1}{2}}$ is calculated with $\mathbf{x_1}$ and so on. Figure 3.5 shows how this is done. Unfortunately we don't know the velocity at $\mathbf{v}_{\frac{1}{2}}$ and usually this is solved by using a different integration method to find this value from the known initial values at $\mathbf{v_0}$.

$$\begin{pmatrix} \mathbf{x_1} \\ \mathbf{v}_{1\frac{1}{2}} \end{pmatrix} = \begin{pmatrix} \mathbf{x_0} \\ \mathbf{v}_{\frac{1}{2}} \end{pmatrix} + h \begin{pmatrix} \mathbf{v}_{\frac{1}{2}} \\ \mathbf{M}^{-1}\mathbf{f}(\mathbf{x_0}, \mathbf{v}_{\frac{1}{2}}) \end{pmatrix} \tag{3.17}$$

For lazy leapfrog we ignore this extra preparing step by assuming the initial value $\mathbf{v}_{\frac{1}{2}}$ is the same as $\mathbf{v_1}$. Normal leapfrog calculates the next position $\mathbf{x_{i+1}}$ using the velocity $\mathbf{v}_{i+\frac{1}{2}}$, half a time step behind. Now we are using the velocity $\mathbf{v_{i+1}}$, the velocity at the same time as the position we are trying to calculate. For applications where accuracy is not too important, this is no problem. Essentially, the only thing we do differently is that we are using a slightly different initial value for $\mathbf{v}_{\frac{1}{2}}$, for the rest lazy leapfrog is equal to the leapfrog procedure.

As far as implementation concerns, if an explicit Euler integrator as previously described is implemented and only the latest values for $\mathbf{x}$ and $\mathbf{v}$ are stored, switching the order in which the position and the velocity is calculated is enough to move from explicit Euler to lazy leapfrog.

- Calculate the forces $\mathbf{f}(\mathbf{x_i}, \mathbf{v_i})$ acting on the particles.
- Update the velocity: $\mathbf{v_{i+1}} = \mathbf{v_i} + h\mathbf{M}^{-1}\mathbf{f}(\mathbf{x_i}, \mathbf{v_i})$.
- Update the position: $\mathbf{x_{i+1}} = \mathbf{x_i} + h\mathbf{v_{i+1}}$.

3.5.3 Implicit Euler integration

Using implicit integration in mass-spring models is a bit more challenging than using explicit integration. Implicit integration methods lead to non-linear equations which require more effort to solve. Recall that the difference between the methods is that the implicit method relies on values from the same time step instead of values from the previous time step. So the change of velocity and position over a time step h is based on the velocity and the position we are actually trying to calculate. We need to make some adjustments to (3.15) but we can not simply replace $\mathbf{x_0}$ by $\mathbf{x_1}$ and $\mathbf{v_0}$ by $\mathbf{v_1}$ since at this point they are still unknown. Equation (3.14)

gives us an alternative. Every entry of $\mathbf{v}_0$ is replaced by $\mathbf{v}_0 + \Delta\mathbf{v}$ and every entry of x_0 is replaced by $\mathbf{x}_0 + \Delta\mathbf{x}$ resulting in (3.18).

$$\begin{pmatrix} \Delta\mathbf{x} \\ \Delta\mathbf{v} \end{pmatrix} = h \begin{pmatrix} \mathbf{v}_0 + \Delta\mathbf{v} \\ \mathbf{M}^{-1}\mathbf{f}(\mathbf{x}_0 + \Delta\mathbf{x}, \mathbf{v}_0 + \Delta\mathbf{v}) \end{pmatrix} \tag{3.18}$$

Having $\Delta\mathbf{v}$ on both sides of equation (3.18) is not practical if we want to find a solution for $\Delta\mathbf{v}$. Here we present an explanation of the solution proposed in [BW98]. To read the whole derivation in detail one should look into that paper. They start by focusing on the second row of (3.18) and state that we need to make two substitutions to set a first step towards finding a solution for $\Delta\mathbf{v}$. The first substitution (3.19) introduces two gradients by which the forces $\mathbf{f_1}$ can be calculated.

$$\mathbf{f}(\mathbf{x}_0 + \Delta\mathbf{x}, \mathbf{v}_0 + \Delta\mathbf{v}) = \mathbf{f}_0 + \frac{\partial \mathbf{f}}{\mathbf{x}}\Delta\mathbf{x} + \frac{\partial \mathbf{f}}{\partial \mathbf{v}}\Delta\mathbf{v} \tag{3.19}$$

These gradients are an important part of the implicit integrator and they must be known at each time step. Equation (3.20) shows an example of the partial derivative of a n-dimensional vector $\mathbf{p}$ over a n-dimensional vector $\mathbf{q}$ for those who are not too familiar with partial derivatives in linear equations. The gradient $\frac{\partial \mathbf{f}}{\partial \mathbf{x}}$ is an $3n \times 3n$ matrix, or a more intuitive description, an $n \times n$ matrix of 3×3 submatrices or blocks. If $\mathbf{K}_{ij}$ is the submatrix at the i-th block-row in the j-th block-column, then $\mathbf{K}_{ij}$ describes the change of force on particle i, in case particle j is moving. Similar for $\frac{\partial \mathbf{f}}{\partial \mathbf{v}}$, when $\mathbf{L}_{ij}$ is the submatrix at the i-th block-row in the j-th block-column, then $\mathbf{L}_{ij}$ describes the change of force on particle i, in case the velocity of particle j changes.

These gradients are an important part of the implicit integrator and they must be known at each time step. Equation (3.20) shows an example of the partial derivative of a n-dimensional vector $\mathbf{p}$ over a n-dimensional vector $\mathbf{q}$ for those who are not too familiar with partial derivatives in linear equations. The gradient $\frac{\partial \mathbf{f}}{\partial \mathbf{x}}$ is an $3n \times 3n$ matrix. Perhaps a more suitable description, a $n \times n$ matrix of 3×3 submatrices or blocks. Let $\mathbf{K}_{ij}$ be the submatrix at the i-th block-row in the j-th block-column, then $\mathbf{K}_{ij}$ describes the change of force on particle i, in case particle j is moving. Similar for $\frac{\partial \mathbf{f}}{\partial \mathbf{v}}$, when $\mathbf{L}_{ij}$ is the submatrix at the i-th block-row in the j-th block-column, then $\mathbf{L}_{ij}$ describes the change of force on particle i, in case the velocity of particle j changes.

$$\frac{\partial \mathbf{p}}{\partial \mathbf{q}} = \begin{pmatrix} \frac{\partial p_1}{\partial q_1} & \frac{\partial p_1}{\partial q_2} & \cdots & \frac{\partial p_1}{\partial q_n} \\ \frac{\partial p_2}{\partial q_1} & \frac{\partial p_2}{\partial q_2} & \cdots & \frac{\partial p_2}{\partial q_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial p_n}{\partial q_1} & \frac{\partial p_n}{\partial q_2} & \cdots & \frac{\partial p_n}{\partial q_n} \end{pmatrix} \tag{3.20}$$

For every spring, some blocks must be inserted in both gradient field matrices. In [BW98] this is explained by the use of constraints, but in our opinion this unnecessarily complicates things for a simple mass-spring model. Therefore the explanation in here is a little different.

When a particle moves, it may change the length of the spring it is connected to thereby influencing the force acting on itself and the particle at the other end of the spring (3.8). For example, assume that there is a spring between particle 1 and particle 2 and that $\hat{\mathbf{n}}$ is the unit direction of the spring from particle 1 to particle 2. Now particle 2 is moved slightly while we observe how the force on this particle changes. Only when it is moved parallel to $\hat{\mathbf{n}}$ a change in spring length will occur, accompanied by a change in force. For particle 2, moving away from particle 1 leads to a change in force towards particle 1 of $-k\hat{\mathbf{n}}$. Moving towards particle 1 leads to a force away from particle 1 of $k\hat{\mathbf{n}}$. The submatrix entry $\mathbf{K}_{22}$ in $\partial\mathbf{f}/\partial\mathbf{x}$ then becomes:

$$\mathbf{K}_{22} = -k \begin{pmatrix} \hat{n}_x & 0 & 0 \\ 0 & \hat{n}_y & 0 \\ 0 & 0 & \hat{n}_z \end{pmatrix} \tag{3.21}$$

Like in (3.8), the effect of moving particle 2 for the other particle 1 is opposite to that of particle 2:

$$\mathbf{K}_{12} = k \begin{pmatrix} \hat{n}_x & 0 & 0 \\ 0 & \hat{n}_y & 0 \\ 0 & 0 & \hat{n}_z \end{pmatrix} \tag{3.22}$$

When we look at the changes in force when particle 1 starts to move around, we can see that $\mathbf{K}_{11} = \mathbf{K}_{22}$ and $\mathbf{K}_{21} = \mathbf{K}_{12}$. For all sets of particles that are not interacting through a spring the corresponding entries remain 0.

The block entries for the field gradient $\frac{\partial \mathbf{f}}{\partial \mathbf{v}}$ are very similar to those for $\frac{\partial \mathbf{f}}{\partial \mathbf{x}}$. This gradient field also works only in the direction of the string (3.9). A change of velocity in the direction of the spring gives a corresponding change in damping force. For the spring-connected particles 1 and 2 we have the following submatrices:

$$\begin{aligned} \mathbf{L}_{11} &= \mathbf{L}_{22} = -b \begin{pmatrix} \hat{n}_x & 0 & 0 \\ 0 & \hat{n}_y & 0 \\ 0 & 0 & \hat{n}_z \end{pmatrix} \\ \mathbf{L}_{12} &= \mathbf{L}_{21} = b \begin{pmatrix} \hat{n}_x & 0 & 0 \\ 0 & \hat{n}_y & 0 \\ 0 & 0 & \hat{n}_z \end{pmatrix} \end{aligned} \tag{3.23}$$

The reader might have noticed that both $\frac{\partial \mathbf{f}}{\partial \mathbf{x}}$ and $\frac{\partial \mathbf{f}}{\partial \mathbf{v}}$ are symmetrical.

Continuing on the subject of gradients later, we first apply the substitution (3.19) in the second row of (3.18):

$$\Delta \mathbf{v} = h\mathbf{M}^{-1} \left(\mathbf{f}_0 + \frac{\partial \mathbf{f}}{\mathbf{x}} \Delta \mathbf{x} + \frac{\partial \mathbf{f}}{\partial \mathbf{v}} \Delta \mathbf{v} \right). \tag{3.24}$$

Next, the term $\Delta \mathbf{x}$ is replaced by $h(\mathbf{v}_0 + \Delta \mathbf{v})$:

$$\Delta \mathbf{v} = h\mathbf{M}^{-1} \left(\mathbf{f}_0 + \frac{\partial \mathbf{f}}{\mathbf{x}} h(\mathbf{v}_0 + \Delta \mathbf{v}) + \frac{\partial \mathbf{f}}{\partial \mathbf{v}} \Delta \mathbf{v} \right). \tag{3.25}$$

Regrouping an multiplying by M eventually leads to a linear system, where the only unknown is the vector $\Delta \mathbf{v}$:

$$\left(\mathbf{M} - h\frac{\partial \mathbf{f}}{\partial \mathbf{v}} - h^2 \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \right) \Delta \mathbf{v} = h \left(\mathbf{f}_0 + h\frac{\partial \mathbf{f}}{\partial \mathbf{x}} \mathbf{v}_0 \right) \tag{3.26}$$

The matrix in the linear system here is large, sparse, symmetric and positive semi-definite. To solve a system where such a matrix is involved, Baraff and Witkin suggested in [BW98] to use a (preconditioned) conjugate gradient solver, which is indeed well suited for the job. An explanation about the inner workings of the conjugate gradient solver can be found in [BF05]. Barraf and Witkin's solver is modified to support additional constraint support, but the main idea behind the conjugate gradient solver remained.

Now that finding $\mathbf{v}_1$ is taken care of, we can use it to find $\mathbf{x}_1$ (3.27).

$$\mathbf{x}_{i+1} = \mathbf{x}_i + h\mathbf{v}_{i+1} \tag{3.27}$$

After we have found $\mathbf{v}_1$ and $\mathbf{x}_1$ both, the whole process starts over again. Every iteration the following steps are taken:

- Calculate the forces $\mathbf{f}(\mathbf{x}_i, \mathbf{v}_i)$ acting on the particles.
- Fill the matrices for the force gradients.
- Construct the linear system which needs to be solved to find the velocity.
- Solve the linear system to find the velocity $\mathbf{v}_{i+1}$.
- Update the position: $\mathbf{x}_{i+1} = \mathbf{x}_i + h\mathbf{v}_{i+1}$.

3.6 Summary

This chapter has been about numerical integration in general but especially about its application in point-mass based physics simulation. First we derived first order Euler integration methods from a Taylor series and discussed some of the differences between explicit and implicit methods. Next there was an explanation about stiff equations and how one can choose the best integration method in such cases. We came to the conclusion that for both first order Euler methods, the implicit integrator is the best choice for stiff equations. Preceding the discussion on point-mass integration we introduced some aspects of the mass-spring model, so that we could use it as a reference model. In the last section we demonstrated how the mass-spring model could be integrated over time using explicit Euler integration, leapfrog integration and implicit Euler integration.

4 The CMPM

The CMPM or Continuum Mechanics Particle Model, was the main subject of Everts' final thesis. The model is based on a pairwise particle interaction combined with elementary continuum mechanics to create a model that is both close to the mathematical understanding of materials and intuitive. The model is strongly related to mass-spring models in the sense that it uses pairwise interactions close to those of mass-spring models. What makes this model interesting is that it distinguishes between tensile strains and shear strains, but there is a side note on that. Also the spring constants are uniquely determined for each interaction pair and based on CM-principles.

In this paper the contribution to the original model is the implementation of the implicit Euler integrator, replacing the leap frog implementation of Everts' [Eve06]. This significantly decreases the number of time steps needed. Then we discuss where the model fails and how unintentionally anisotropy is introduced. As it turned out, the CMPM is actually based on false assumptions and therefore simulations do not demonstrate the intended results. A couple of attempts to eliminate the requirement that the Voronoi graph of the undeformed particle configuration is known are described. So far this has seemed to be ambitious and has not led to satisfying results.

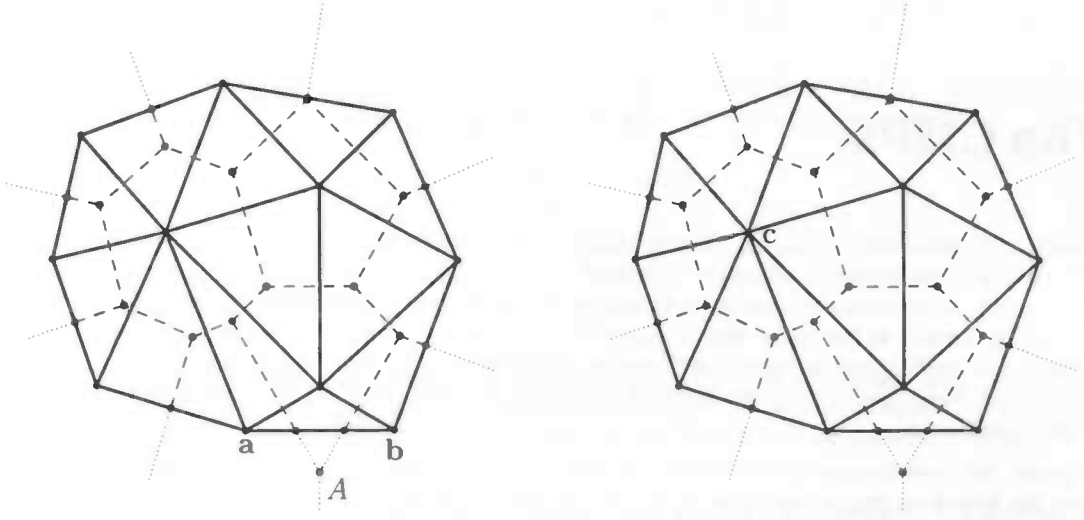
4.1 CMPM introduction

Much like a mass-spring model, the CMPM relies on point masses or particles, pairwise interacting with each other. The method on which the connection scheme is based is a tetrahedrisation of the deformable body. At the location of every vertex a particle is positioned. Particles that share an edge form an interaction pair, except some edges lying on the outer hull, but more on that later. By some spring-like mechanism that will be discussed later, the simulated body gets its coherence and makes it tend to return to its undeformed shape when deformed.

The CMPM requires a Delaunay tetrahedralized volume to work with. Though the Delaunay property may be lost under deformation, when the body is undeformed the property must apply. The tetrahedrisation is Delaunay when for every tetrahedron the sphere through the vertices does not encapsulate any other vertices. This called the 'empty sphere' property. Some other interesting properties of the Delaunay tetrahedrisation are that the circumsphere of any tetrahedron is empty as well, the angles between the edges of the tetrahedra are maximized and the boundary of the exterior faces describe the boundary of the convex hull. The main reason that Delaunay tetrahedrisation is a requirement though, is that it has a dual called a Voronoi graph. Over the ins and outs of two dimensional Voronoi graphs and Delaunay triangulations we refer to [dBvKOS97].

Assigning every point in space to the nearest vertex is a method to find the Voronoi graph. Figure 4.1(a) shows an example of a two dimensional Voronoi graph in a triangulation, the two dimensional equivalent of a tetrahedrisation. In the two dimensional case one can see that the group of points between two vertices form a line-border. The infinitely long version of the border splits the space in two areas, where each area contains only those points that are closest to the vertex in that area. In three dimensions these borders are flat polygons, which we will call Voronoi faces. Every pair of particles that share an edge in the tetrahedrisation will have a Voronoi face lying in between them in the Voronoi graph. The plane defined by the Voronoi face is orientated perpendicular to this edge and crosses the edge in the middle. For the CMPM it is rather inconvenient that these Voronoi faces may have an infinitely large area for pairs of vertices near the outer hull, since the force calculation depends on this area later on. To solve this we clip the Voronoi faces that cross the outer tetrahedral faces. Some Voronoi faces may be clipped as a whole and will be ignored. In figure 4.1(a), particles a and b are not interacting in the model since the Voronoi border A is clipped entirely.

The Voronoi cells that are formed around the particles, combined with the outer tetrahedral faces are a useful tool for distributing mass over the particles. Every cell gives us exact information to which particle a volume



(a) Voronoi clipping (b) Particle mass distribution

Figure 4.1: Voronoi graph. The solid lines represent the triangulation, the dotted and dashed lines are part of the Voronoi graph. In the model, the dots connecting the solid lines represent particles. The dots lying on dotted or dashed lines we refer to as Voronoi vertex. The dotted parts of the Voronoi graph are being clipped in the CMPM.

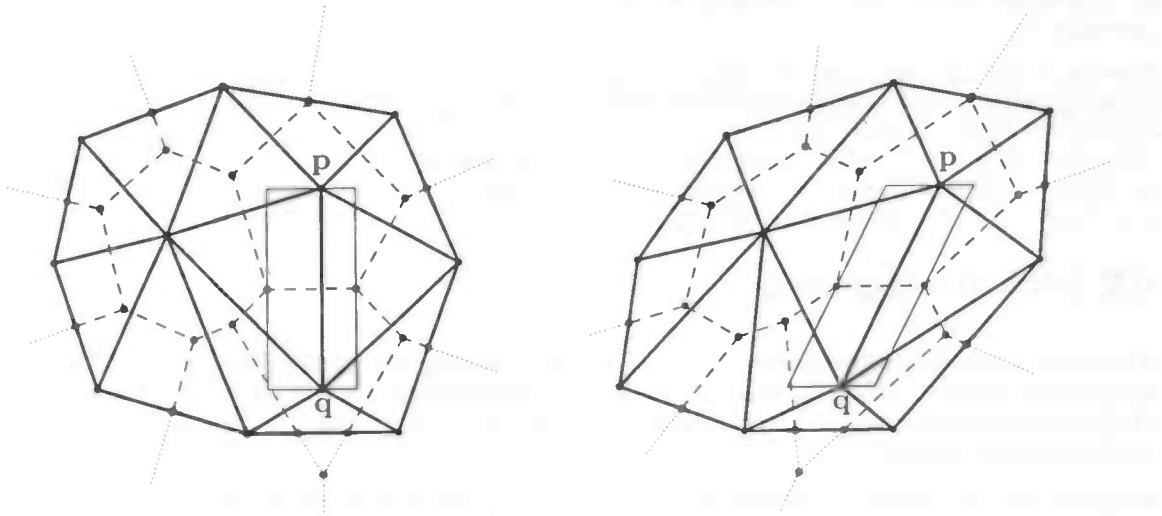
element of material is closest to. This way, the volume of a cell times the density of the material determines the mass of the particle the cell is associated with. In figure 4.1(b), the area that is assigned to particle c is filled gray.

The Voronoi faces that are not entirely clipped form the interacting bond between particles. This explains why some particles lying on the outside of the volume are not connected despite the edge between them. To visualize how the interaction method relates to continuum mechanics one could imagine a beam-like shape, parallel to the edge and bounded by the form of the Voronoi face between the particles in undeformed state. The planes which form the boundary of the shape at the top and the bottom intersect the particles at both side of the edge and are parallel to the Voronoi face. Figure 4.2(a) shows what this looks like in two dimensions.

In section 2.3 we explained the effects of material properties using the example of a cylinder. The application of Young's modulus and the shear modulus is the basis for interaction between two particles in the CMPM. Every cycle and for each interaction pair, the shape of the cylinder is examined to see if deformation has taken place. Tensile and shearing deformations are detected by comparing the position of the two particle to each other in relation to the normal direction of the Voronoi face lying between them (figure 4.2(b)).

During the simulation the Voronoi graph deforms with the model. The position of every Voronoi vertex is stored in barycentric coordinates relative to the four particles representing the tetrahedron in where the vertex is located in undeformed state. With these coordinates, the positions of the Voronoi vertices can be found from particle positions. Though this allows us to recover the graph under deformation, it is no longer possesses the properties of a Voronoi graph. What we do have is a weighted combination of local particles describing a face that is perpendicular to the edge it is related to in case of no strain. The combination of a line parallel to the edge and the plane through the pseudo Voronoi face is what we use to approximate the strain for the volume between the two particles.

When the strain is not uniform, the vertices of the pseudo Voronoi faces may be non-planar. A simple algorithm finds an approximation normal direction for this polygon. The n Voronoi vertices are sorted clockwise around the edge (in simulation preprocessing). Using the center of the edge, triangles are formed with every two vertices next to each other in clockwise order. The normalized sum of the cross products is an approximation for the normal $\hat{n}$ of the polygon. On pages 27 and 28 of [Eve06] this algorithm is also explained. Be aware



(a) Undeformed body (b) Deformed body

Figure 4.2: Voronoi graph

that for low quality tetrahedrisations, the Voronoi vertices for some Voronoi faces may not lie around the edge at all, making clockwise sorting around that edge impractical. To use this algorithm, all the Voronoi vertices must lie around the center of the edge for which they are relevant.

Let $\mathbf{p}$ and $\mathbf{q}$ be the positions of two interacting particles. The scalar a denotes the area of the Voronoi face that lies between the particles and the scalar l denotes the distance between the particles in undeformed state. The vector $\mathbf{r}$ is the difference (4.1) between these two vectors. The vector $\hat{\mathbf{n}}$ is the approximation for the normal of the pseudo Voronoi face and is directed toward $\mathbf{q}$. Vector $\mathbf{m}$ is the direction in which the shearing takes place (4.2).

$$\mathbf{r} = \mathbf{q} - \mathbf{p} \tag{4.1}$$

$$\mathbf{m} = \mathbf{r} - (\hat{\mathbf{n}} \cdot \mathbf{r})\hat{\mathbf{n}} \tag{4.2}$$

With equation (4.3) the tensile strain and the shear strain are found. The deformation in the direction $\mathbf{r}$ is treated as tensile strain and the deformation in any other direction is treated as shear strain.

$$\begin{pmatrix} \frac{\partial r_n}{\partial \hat{\mathbf{n}}} + 1 \\ \frac{\partial r_m}{\partial \hat{\mathbf{n}}} \end{pmatrix} = \frac{1}{l} \begin{pmatrix} \hat{\mathbf{n}}^{\mathbf{T}} \\ \hat{\mathbf{m}}^{\mathbf{T}} \end{pmatrix} \mathbf{r} \tag{4.3}$$

Finally the tensile strain is multiplied by Young's modulus E and the shear strain by the shear modulus G to find the strain resisting forces $\mathbf{f}_q$ for the particle at position $\mathbf{q}$. The forces $\mathbf{f}_p$ for the particle at position $\mathbf{p}$ are in the opposite direction.

$$\mathbf{f}_q = a \begin{pmatrix} E\frac{\partial r_n}{\partial \hat{\mathbf{n}}} & G\frac{\partial r_m}{\partial \hat{\mathbf{n}}} \end{pmatrix} \begin{pmatrix} \hat{\mathbf{n}}^{\mathbf{T}} \\ \hat{\mathbf{m}}^{\mathbf{T}} \end{pmatrix} \tag{4.4}$$

This whole method for calculating forces can be considered a double spring method. Mass-spring models handle strain one dimensionally in the direction of the spring. The CMPM is a two dimensional approach. The directions of the two springs are mutual orthogonal and determined by the direction of the pseudo Voronoi face.

Like in the mass-spring model we would like to have some form of damping force on the difference of velocity between two connected particles. Also here we separate between the tensile direction $\hat{\mathbf{n}}$ and the shear direction

$\hat{\mathbf{m}}$. Therefore there is need for two different damping factors, one for tensile damping b_n and one for shear damping b_m.

When $\mathbf{v}_i$ is the velocity of particle i, the damping forces $\mathbf{f}_q$ in the direction of $\hat{\mathbf{n}}$ and $\hat{\mathbf{m}}$ are given by (4.5). The forces $\mathbf{f}_p$ for the particle at position p on the other side of the edge are in the opposite direction.

$$\begin{aligned} \mathbf{f}_q \cdot \hat{\mathbf{n}} &= -b_n(\hat{\mathbf{n}} \cdot (\mathbf{v}_q - \mathbf{v}_p))\hat{\mathbf{n}} \\ \mathbf{f}_q \cdot \hat{\mathbf{m}} &= -b_m(\hat{\mathbf{m}} \cdot (\mathbf{v}_q - \mathbf{v}_p))\hat{\mathbf{m}} \end{aligned} \tag{4.5}$$

4.2 Implicit Integration

The forces which resists deformation can be very strong, making the differential equations fall in the 'stiff' category (section 3.3). As was explained in the previous chapter, implicit integration methods are the preferred choice to prevent the simulation from becoming unstable. In this section will be demonstrated how the force gradients can be derived.

In section 3.5.3 the implicit Euler integrator for interacting point masses was discussed. For the mass-spring model we demonstrated how to calculate the forces and force gradients input for the integrator. The way to do this for the CMPM is essentially the same, except that there are now two springs for every interaction pair.

In the previous section the forces for the CMPM were derived. For every set of connected particles there is a spring-like force in the direction $\hat{\mathbf{n}}$ and $\hat{\mathbf{m}}$. All we have to do is for every interacting set of particles i and j calculate the value for the relevant submatrices $\mathbf{K}$ in $\frac{\partial \mathbf{f}}{\partial \mathbf{x}}$ and $\mathbf{L}$ in $\frac{\partial \mathbf{f}}{\partial \mathbf{v}}$. This comes down to taking the sum of the force gradients of the two separate springs that connect the two particles (4.6).

$$\begin{aligned} \mathbf{K}_{11} = \mathbf{K}_{22} &= -E\begin{pmatrix} \hat{n}_x & 0 & 0 \\ 0 & \hat{n}_y & 0 \\ 0 & 0 & \hat{n}_z \end{pmatrix} - G\begin{pmatrix} \hat{m}_x & 0 & 0 \\ 0 & \hat{m}_y & 0 \\ 0 & 0 & \hat{m}_z \end{pmatrix} \\ \mathbf{K}_{12} = \mathbf{K}_{21} &= E\begin{pmatrix} \hat{n}_x & 0 & 0 \\ 0 & \hat{n}_y & 0 \\ 0 & 0 & \hat{n}_z \end{pmatrix} + G\begin{pmatrix} \hat{m}_x & 0 & 0 \\ 0 & \hat{m}_y & 0 \\ 0 & 0 & \hat{m}_z \end{pmatrix} \end{aligned} \tag{4.6}$$

For the damping force gradient $\frac{\partial \mathbf{f}}{\partial \mathbf{v}}$ the calculation of values in submatrices $\mathbf{L}$ are also just the sum of the gradients of the two separate springs (4.7).

$$\begin{aligned} \mathbf{L}_{11} = \mathbf{L}_{22} &= -b_n\begin{pmatrix} \hat{n}_x & 0 & 0 \\ 0 & \hat{n}_y & 0 \\ 0 & 0 & \hat{n}_z \end{pmatrix} - b_m\begin{pmatrix} \hat{m}_x & 0 & 0 \\ 0 & \hat{m}_y & 0 \\ 0 & 0 & \hat{m}_z \end{pmatrix} \\ \mathbf{L}_{12} = \mathbf{L}_{21} &= b_n\begin{pmatrix} \hat{n}_x & 0 & 0 \\ 0 & \hat{n}_y & 0 \\ 0 & 0 & \hat{n}_z \end{pmatrix} + b_m\begin{pmatrix} \hat{m}_x & 0 & 0 \\ 0 & \hat{m}_y & 0 \\ 0 & 0 & \hat{m}_z \end{pmatrix} \end{aligned} \tag{4.7}$$

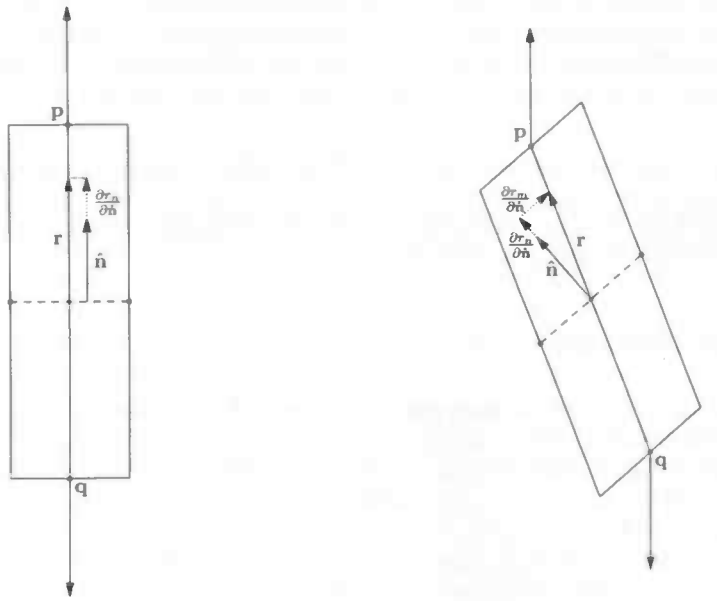
With the introduction of the implicit integrator, the model performance made a large jump. For our average model, the time step could be increased a hundredfold from around 1×10^{-4} to 1×10^{-2}. Because there is work to be done in every step, like solving a large linear system, the simulation rate was increased by 'just' around a factor 20 or 30.

4.3 Unintended anisotropy

Because the tensile strain and shear strain components are handled separately, one may expect that the behaviour of a volume under CMPM is correct with respect to those two types of deformation. Young's modulus should come into effect under tensile deformation, the shear modulus should under shear deformation. Simulation tests do not show this to be the case. As long as tensile deformation takes place perpendicular to the

Voronoi face and shear deformation parallel to the Voronoi face, there are no problems. However when strain acts in different direction, things go wrong.

Let us see what is going wrong. Consider a single edge with its Voronoi face, projected onto a plane orthogonal to the Voronoi face. Figure 4.3(a) shows how this model behaves well under the described specific circumstances. The volume is stretched vertically and the tensile deformation correctly turns up at the edge. Figure 4.3(b) next to it shows the same edge rotated 30 degrees. Again the volume is stretched vertically as in figure 4.3(a), but nevertheless the tensile strain found at the edge is in a whole other direction and a shear strain has appeared as well.



(a) No rotation (b) Rotation of 30 degrees

Figure 4.3: An edge under tensile strain

Looking at the box of material around the edge and the Voronoi face in figure 4.3(b), it is no surprise that a shear tensor appears since the box is actually sheared. Depending on the orientation of the edge the CMPM detects a different mix of tensile and shear strains. Though the local behaviour is not what is expected, one might hope that non-parallel edges cancel forces out in such that the model globally still works correctly. This is not the case as one could come up simple situations where the global behaviour is just as bad. One of the features of this model is that Young's modulus and the shear modulus can be set separately, but this is also on of the reasons there may be a big difference in edge response when it is stretched on way or the other.

That the edges display different response under different orientations has another consequence. Under a similar deformation one expects the same response in an isotropic material (section 2.4), but for the CMPM this is also not true. The tetrahedra and in fact the whole model demonstrates anisotropy.

Sadly one must conclude that strain decomposition like in CMPM can not be justified by arguments of more realistic material behaviour. By splitting the strain up and allowing a user to set two separate elasticity constants does provide the user with more control than there is with the mass-spring model. It is not certain what the exact effect of each of the elasticity values is. Suggesting that the volume during the simulation will behave accordingly to the values of Young's modulus and the shear modulus is deceiving. What makes CMPM significantly better than the basic mass-spring model remains to be seen.

4.4 Eliminating the Voronoi graph

This section is about some attempts to eliminate the need for a Voronoi graph from the CMPM. Note that we did not know about the anisotropic behaviour of the CMPM (section 4.3) until after this research. Otherwise we might have saved ourselves the trouble of looking into this too deeply.

Though the Voronoi graph is entirely constructed in the preprocessing phase, there are still reasons why one would want to get rid of the Voronoi graph. First, calculating the Voronoi graph is not an easy thing to do, especially in three dimensions. This results in the preprocessing phase being many times more complex than the force calculation method. A model where the preprocessing has a lesser chance to scare people away would be nice. Second, in cases where you would want the material to tear under strong forces, apply cuts or take other permanent deforming actions on the simulated body, the Voronoi graph needs to be re-evaluated. Do-able as it might be, this pulls the difficult Voronoi graph construction from the pre-processing phase into the simulation phase introducing all kinds of new problems. Third, the requirements for a Delaunay tetrahedrisation are because the dependence of the CMPM on a Voronoi graph. Without a Voronoi graph such requirements might be loosened a bit.

Our thought was to circumvent these difficulties by simply eliminating the need for the Voronoi graph. The (pseudo) Voronoi faces of the graph are used as a reference for the material context perpendicular to the edge between two connected particles, in order to find shear strain. Another method to find the direction for tensile strain would make the Voronoi graph redundant.

4.4.1 Barycentric subfaces

The first attempt was to replace the Voronoi graph by a simpler version of the graph. Imagine the case of the perfect tetrahedrisation where every tetrahedron has edges which are all of equal length. Every tetrahedron would also have one Voronoi point located in its center. The idea was to make this also the case for not so perfect tetrahedrisations. Every tetrahedron will contain six subfaces. The vertices of each subface is determined by the barycentric center of the tetrahedron, the center of one of the edges and the barycentric center of the two tetrahedral faces connected to that edge. Figure 4.4 shows how a tetrahedron with these subfaces. The Voronoi face replacement for an edge is the combination of subfaces that are connected to the edge. The subfaces are to be reconstructed at every iteration and the unit vector $\hat{\mathbf{n}}$ of the sum of the normals of the subfaces represents the direction perpendicular to the subfaces and determines the direction for tensile strain for the edge.

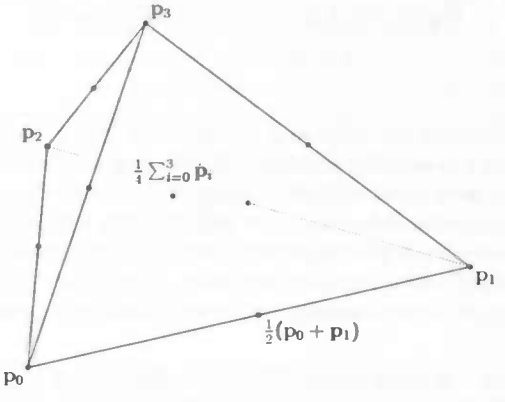


Figure 4.4: A tetrahedron with six subfaces

When the edges of the tetrahedron are not of the same size, initially the vector $\hat{\mathbf{n}}$ does not always point in the same direction as the corresponding edge. If we would calculate the forces the same way as in section (4.1), then there would be a shear force without any deformation. To fix this we decided to compare $\hat{\mathbf{n}}$ no longer to the direction of the edge itself, but to the direction of $\hat{\mathbf{n}}$ in the undeformed body. The value of $\hat{\mathbf{n}}$ at $t = 0$ is stored in barycentric coordinates, relative to one of the vertices of the tetrahedron, and recovered as $\mathbf{r}$

for every iteration. Because it is stored in barycentric coordinates, this vector $\mathbf{r}$ is obviously influenced by the deformation of the tetrahedron. For the direction of shear strain $\mathbf{m}$, we calculate (4.8) the orthogonal difference between $\hat{\mathbf{n}}$ and $\mathbf{r}$.

$$\mathbf{m} = \mathbf{r} - (\hat{\mathbf{n}} \cdot \mathbf{r})\hat{\mathbf{n}} \tag{4.8}$$

The strain is a little easier to find (4.9) than in (4.3) because there is no need to divide by the length of the edge.

$$\begin{pmatrix} \frac{\partial r_n}{\partial \hat{\mathbf{n}}} \\ \frac{\partial r_m}{\partial \hat{\mathbf{n}}} \end{pmatrix} = \begin{pmatrix} \hat{\mathbf{n}}^T \\ \hat{\mathbf{m}}^T \end{pmatrix} (\mathbf{r} - \mathbf{n}) \tag{4.9}$$

The stress force is then calculated (4.10) with the strain. The area for edge a used here is simply the sum of the areas of the subfaces belonging to that edge in the undeformed body. The direction of the counteracting force depends on the direction of $\hat{\mathbf{n}}$. When it belongs to an edge described by particles $\mathbf{p}$ and $\mathbf{q}$ and $\hat{\mathbf{n}}$ points in the direction from $\mathbf{p}$ to $\mathbf{q}$, the force calculated in this way is the acting stress force on particle $\mathbf{p}$, or the counteracting force on particle $\mathbf{q}$.

$$\mathbf{f} = a \begin{pmatrix} E\frac{\partial r_n}{\partial \hat{\mathbf{n}}} & G\frac{\partial r_m}{\partial \hat{\mathbf{n}}} \end{pmatrix} \begin{pmatrix} \hat{\mathbf{n}}^T \\ \hat{\mathbf{m}}^T \end{pmatrix} \tag{4.10}$$

Maarten Everts and I both worked on an implementation of this method. In his paper [Eve06] it is referred to as the CMPM2. There are some benefits to this method, probably the most important one is that it is one of the simplest ways to replace the Voronoi faces. It also corrects automatically some of the instabilities in the Voronoi method since it is able to run using a bit larger time steps. An important downside of the CMPM that remains is that the way that tensile strain and shear strain are distinguished does not lead to good results.

We will give an example of one of the tests we performed on the models to see if the forces were calculated correctly. The example is in two dimensions, but the principles are analogous to three dimensional cases. We are going to look at the tensile and shear strain in a triangle, calculated as in (4.9). Note that though we look a single triangle, we consider it to be part of a larger body. Under deformation we compare the vector $\mathbf{r}$ and the vector $\hat{\mathbf{n}}$, which is the current normal vector of the subfaces.

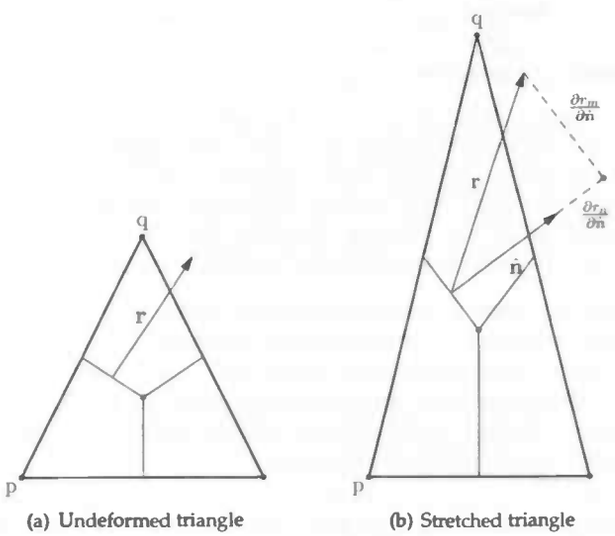


Figure 4.5: Example of calculated forces with barycentric subfaces.

Figure 4.5 shows left the undeformed triangle and right the deformed triangle, which is located somewhere inside a body. What one would expect if a triangle is stretched in such a way, is that the counteracting force on particle $\mathbf{q}$ is largely based on stress resulting from tensile strain. However if we look at on of the edges connected to particle $\mathbf{q}$ in the right picture and plot the shear and tensile strain components this method comes up with, we see that it returns a small tensile strain and a much larger shear strain. These results are like our findings described in section 4.3.

4.4.2 Best fitting plane

Each edge in a tetrahedrisation belongs to at least one or more tetrahedra. A tetrahedron has four vertices, so besides the two vertices belonging to the edge there are two other vertices. The set of unique vertices belonging to tetrahedra that all have on edge in common minus the vertices that determine the concerned edge, form roughly a ring surrounding the edge (figure 4.6). A plane fitted to these points would in many cases be roughly perpendicular to the edge. Under deformation we might determine the direction for tensile strain in a similar way using the normal vector of this plane as we do in the CMPM using Voronoi faces.

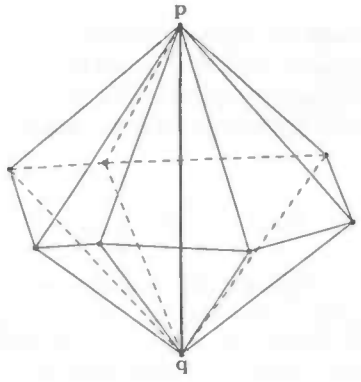


Figure 4.6: An edge, in this case surrounded by 7 particles

First we figured out how close these planes are to perpendicular with respect to the edge they belong to. The Voronoi faces were exactly perpendicular to the edge and if the planes were not, the behaviour of the model would differ from the CMPM's. The best fitting plane through these points can be found using a least square error method. This requires us to find the eigenvectors of a 3×3 matrix, which is computationally too expensive to do for every edge every time iteration. It could be used to explore the feasibility of finding a perpendicular plane out of the ring of particles surrounding an edge.

When the least square error approach would give good results, a computationally low cost way to find the normal of the plane might also work. For instance, my supervisor suggested that every ring of particles could be split by a plane to form two sets of particles. For both groups we would calculate the mean position and calculate the vector between the two sets. Now we split the ring of particles with a different plane so we have two other sets of particles from which we calculate ourselves another vector. The cross product of these two vectors might be good enough to represent the normal of the plane through the ring of particles.

The more particles in the ring the better since this reduces the influence of a one particle being greatly out of line with the others and other anomalies. For a high quality tetrahedrisation there should on average be about six particles in the ring, except for some edges on the outer hull. Admittedly this is already a small number to work with. We examined a volume that had been commonly used for all kinds CMPM related experiments. It was a beam shape with a hole through and contained 1679 particles, 7068 tetrahedra and 9835 edges. For every edge we counted by how many particles it was surrounded.

Particles	2	3	4	5	6	7	8	9	10	11	12
Edge	140	1386	2091	1874	1820	1449	753	250	60	10	2

As expected there where between five and six particles in each ring on average. There also appeared to be a number of edges on the outer hull with only two particles related to them, not enough to find a unique plane for. For now we ignored this and moved on to find the quality of the normal vectors of the planes we would find using the least square error method, except for edges with three particles for which we used the cross product. Edges with two particles were left out, but for later implementations we used the centre of the edge so that combined with the two particles we could find a plane. For the remaining edges we stored the angle between the edge and the plane normal vector. Figure 4.7 shows the distribution of these angles.

Many of the planes seemed to be far from perpendicular to the edge. We were working towards an alternative model that still closely resembled the original CMPM model, so we preferred the perpendicular planes. Many varying angles meant that the behaviour might become unpredictable in terms of stability. We had already experienced that the performance during simulation depends on the weakest link. One odd behaving pair of particles could ruin the simulation.

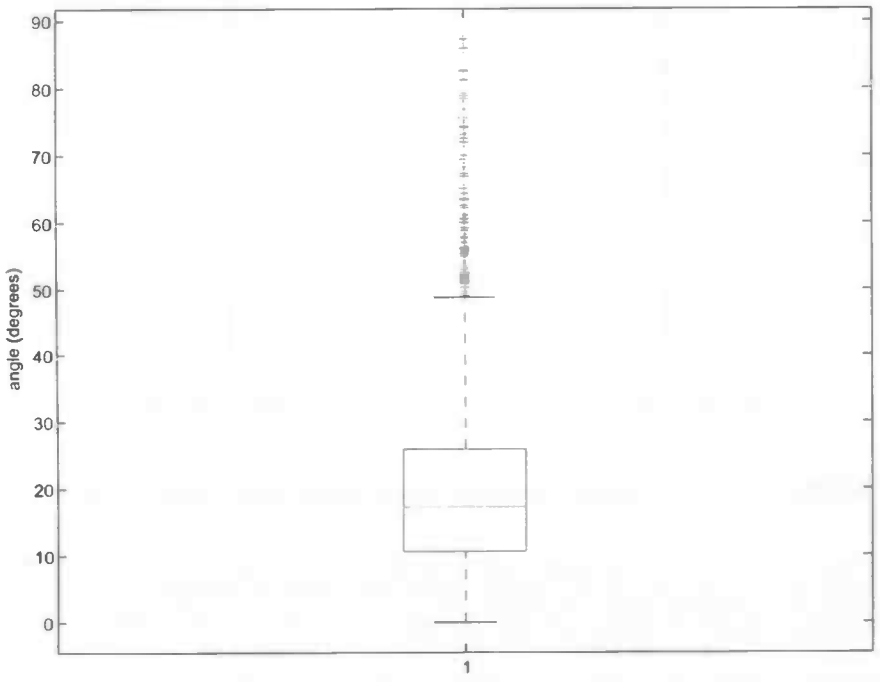


Figure 4.7: Boxplot for the angles of the best fitting plane

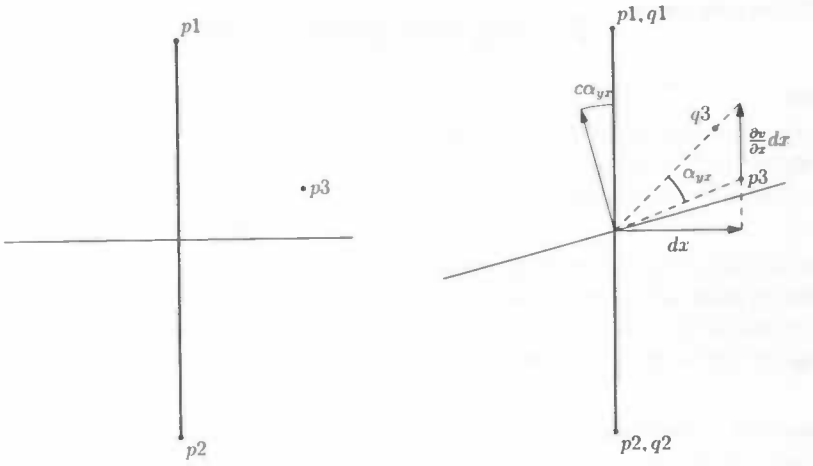
An implementation where the best fitting plane indicated the direction for tensile strain did not impress. Obviously the principle component analysis for every edge reduced the simulation speed considerably. The model was also less stable than CMPM so that measures had to be taken to lessen the reaction to stress for every time step by changing material and environmental constants (gravity) and by using a smaller time step. With no easy way, if there is any, to fix these shortcomings, we abandoned the best fitting plane approach.

4.4.3 Tilted plane

An original way to find an alternative Voronoi face was the one of the tilted plane. It is based on the same edge particle relation as for the best fitting plane method, where the direction of the tensile strain was determined by the particles surrounding the edge. See section 4.4.2 for what we mean with surrounding particles. The idea was that a plane, perpendicular to the edge in undeformed state, would tilt away from the right angle when particles moved with respect to the edge. The most difficult part was to find a sound explanation how particle movement would affect the angle of the plane. This method was never implemented because the many possible particle configurations around an edge made the whole unlikely to produce stable results, similar to the best fitting plane method.

In figure 4.8 we have an edge $\mathbf{p_1p_2}$ with one of its surrounding particles at $\mathbf{p_3}$. Note that the surrounding particles do not necessarily have to be positioned on the plane, but could be located above or below. The tilted plane in figure 4.8(a) with the body in undeformed state is perpendicular to the edge and in the direction of the edge centered to the surrounding particles. When in figure 4.8(b) the particle at position $\mathbf{p_3}$ is moved to position $\mathbf{q_3}$, the plane follows by tilting by a factor c times the angle between $\mathbf{p_3}$ and $\mathbf{q_3}$. The axis over which the plane is tilted is the one perpendicular to the edge and the line between $\mathbf{q_3}$ and the point where the edge

and plane intersect. The total angle by which is the plane tilted is the sum of the influence of all particles. The constant c is a weight uniquely determined for each surrounding particle based on the distribution of the particles surrounding the edge.



(a) Undeformed body (b) Deformed body

Figure 4.8: Two-dimensional projections of an edge with the tilted plane.

We will suggest a simple way to find a value c, but first we look into the case where we have an edge, with infinitely many particles surrounding it in a perfect circle. We apply an angular deformation of α to the material, leading to the situation as in figure 4.9. For this case we no longer look at single particles but calculate the angular deformation based on the deformed continuous circle. We only focus on the direction to where the angular deformation takes place. Angles in other directions cancel each other out due to symmetry. The angular deformation is found using (4.11), where b is some constant.

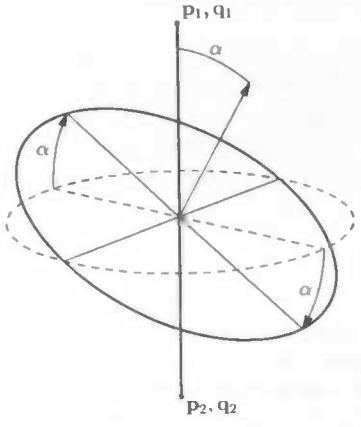


Figure 4.9: Infinitely many surrounding particles in a perfect circle.

$$\alpha b \int_0^{2\pi} |\sin t| dt = 2\alpha b \int_0^{\pi} \sin t dt = 2\alpha b[-\cos t]_0^{\pi} = \alpha \tag{4.11}$$

For this equation to be true we find that $b = \frac{1}{4}$. Now we continue with finding values for c. Consider the case where we have an edge with n particles $\{p_1..p_n\}$ surrounding it. Figure 4.10 gives an impression of such an

edge for $n = 7$, looking in a direction parallel to the edge. For the particles we must find a value for $\{c_1..c_n\}$. First we sort the particles surrounding the edge counter clockwise. Then we calculate the angle of the segments between all subsequent particles. We define β_i as the sum of the half angle of the two neighbouring segments. Then c_i is calculated as in (4.12). This way we let each particle $\mathbf{p}_i$ represent part of the circle in (4.11).

$$c_i = \frac{b}{2\pi}\beta_i \tag{4.12}$$

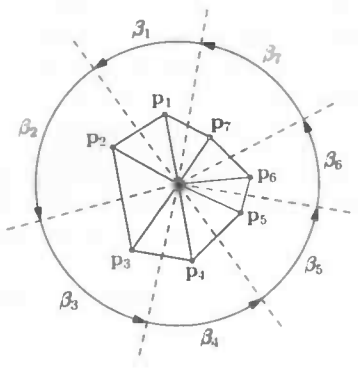


Figure 4.10: Surrounding particles. The edges connecting the particles with the edge in the center are also displayed.

We have no test results for this method because we had abandoned it at an early stage. We figured that the existence of an unstable interaction pair is very likely, considering the small margins wherein this model might be reasonably correct and the infinite number of possible surrounding particle configurations.

4.5 Evaluation

The study on CMPM we have described in this chapter was a continuation of the work of Everts [Eve06]. We were able to improve performance by replacing the explicit integrator with an implicit integrator and increase simulation performance by a significant amount. We also demonstrated that sadly the idea behind the model itself is inherently incorrect. The discussion on the CMPM was concluded with the discussion of three methods to eliminate the Voronoi graph to simplify the model. One method based on barycentric subfaces was the most successful, outperforming the original model in some ways. The other two approaches using the so called best fitting plane and the tilted plane, are likely to be too unreliable to produce stable simulations.

5 Shear by volume preservation

The CMPM model (chapter 4) relied on Young's and the shear Modulus to calculate stress forces from stretching or compressing and shear deformations. In (2.3) we explained that there is a relation between Young's modulus, the shear modulus and Poisson's ratio. In short it was said that a higher Poisson's ratio means more volume preservation and less shear resistance.

So far we had not actively used Poisson's ratio in the CMPM simulations, though when the shear modulus was increased, there was also an increased amount of volume preservation under the same global deformation. We wondered what if instead of trying to use the shear modulus to indirectly enforce volume preservation we try to preserve our volume directly based on Poisson's ratio. When we neglect the shear modulus, will it pop up anyway due to the relation between the material's constants?

In order to find the answer to this question we built a model which is essentially a mass spring model, with a small addition to it to preserve volume. If we would indeed be able to control shear resistance by Poisson's ratio, this would mean we had found another way around the problem we encountered with CMPM. That is, a tetrahedrized volume simulation model, without the need for a Voronoi graph. The force calculation might even proove to be more robust than in the CMPM. It must be mentioned though that this is not an entirely new approach. Only the fact that it is essentially a mass-spring model and the fact that many people build variations on this principle makes it likely that similar implementations have preceeded ours.

5.1 Mass spring with volume preservation

Like with the CMPM, the basis of this model is a volume partitioned by tetrahedra. The mass of the volume is assigned to particles at the position of the vertices of the tetrahedra. A quarter of the mass of a tetrahedron is assigned to each of its vertices. During the simulation the tetrahedron vertices follow the particles' movement. The interaction scheme on which the point masses or particles influence each other is based on the tetrahedrisation of the volume. Every set of particles that share an edge of a tetrahedron becomes an interaction pair. A spring is placed in between them with a spring constant k. A similarity of this model to the CMPM can be found in the way the spring constant k is uniquely calculated for each edge (5.1). Vectors $\mathbf{p}$ and $\mathbf{q}$ denote the positions of particles of an edge. The scalars l and a stand for the length and the cross-sectional area of the edge when the body is undeformed and E is Young's modulus. Like in CMPM each edge represents a cylinder-like volume which here is reduced to a simple spring. Finding the length l is trivial and for an approximation of the area a a similar approach as in 4.4.1 suffices.

$$k = \frac{Ea}{l} \tag{5.1}$$

The unit vector $\hat{\mathbf{n}}$ is the direction from $\mathbf{p}$ to $\mathbf{q}$. The forces for the mass-spring model (3.8) become (5.2).

$$\begin{aligned} \mathbf{f}_p &= Ea\hat{\mathbf{n}}\tfrac{|\mathbf{q}-\mathbf{p}|-l}{l} \\ \mathbf{f}_q &= -Ea\hat{\mathbf{n}}\tfrac{|\mathbf{q}-\mathbf{p}|-l}{l} \end{aligned} \tag{5.2}$$

For the volume preservation part the volume difference is examined for each tetrahedron in every simulation cycle. The volume at that moment in time is compared with the volume of the tetrahedron in undeformed state. When the volume of the tetrahedron has changed, actions are taken to counteract this. An additional force in the direction of the edge is added next to the spring force in order to stimulate the tetrahedron to return to its original volume.

What is left is a way to preserve the tetrahedron volume. We could assume that all the edges of the tetrahedron had the same length b originally. The volume v for such a tetrahedron is well defined (5.3). For a certain

amount of tetrahedron volume change it can be derived what change in edge length is responsible. The volume restoring force on the edges could be proportional to this change in edge length. When we look at Poisson's ratio this may be unnessecarily complicated and even wrong.

$$V = \frac{\sqrt{2}}{12} b^3 \tag{5.3}$$

Poisson's ratio is the rate of transverse contraction strain in response to tensile strain (2.21). If a cube is stretched in a direction perpendicular to two of its opposing faces, the change in volume V is equal to the amount it stretched when there is no volume preservation. In our case we don't know in which direction the tetrahedron is stretched, but we could just assume that the ratio of change in volume represents the strain in some unknown directions that caused the volume of the tetrahedron to change. Without the strain direction we also don't know the transverse directions in which to propagate the strain by the amount specified by Poisson's ratio ν. What we decided to do is propagate it in any direction, over all edges of the tetrahedron. This additional strain combined with the tensile strain on the edge completes the force by which the edge will expand or shrink in this model.

ΔV_i is the change in volume (5.4) of tetrahedron i. The value of $V_{i,0}$ stands for the volume of the undeformed tetrahedron i and V_i denotes the volume of the deformed tetrahedron i.

$$\Delta V_i = V_i - V_{i,0} \tag{5.4}$$

For every edge that lies between particles $\mathbf{p}$ and $\mathbf{q}$, volume preserving forces are added to the particles for every tetrahedron i where the edge appears.

$$\begin{aligned} \mathbf{f}_p &= -Ea\nu \frac{\Delta V_i}{V_{i,0}} \hat{\mathbf{n}} \\ \mathbf{f}_q &= Ea\nu \frac{\Delta V_i}{V_{i,0}} \hat{\mathbf{n}} \end{aligned} \tag{5.5}$$

Besides the mass-spring forces there are now also volume preserving forces. In the next section we will run a test to see if this combination will result in resistance to shear strain.

5.2 Simulation results

We have tested how much shear the volume preservation forces introduce. For this we used the same box object as Everts' used in [Eve06] to test the shear forces in CMPM and CMPM2. The edges of this cube are all of length 1. For integration we used the leapfrog integrator (section 3.5.2). The material settings where as follows: Young's modulus was set to $3 \times 10^6 \frac{N}{m^2}$ and the density of the model was set to $4 \times 10^3 \frac{kg}{m^3}$. We ran two simulations where the cube was sheared until the shear strain had increased from 0 to 1. One simulation was without volume preservation enabled ($\nu = 0$) and the other was with volume preservation enabled ($\nu = 0.3$). For higher values of Poisson's ratio ν, the simulation became somewhat unstable.

First we wanted to know if there was actually less change in volume when Poisson's ratio was larger than 0. Figure 5.1 shows that this is the case. Beyond a shear strain of 0.5 the volume also drops for some reason. That amount of strain is quite extreme and we considered it therefore not a big issue, also because the volume preservation is still better than when Poisson's ratio is 0.

Next we looked at the shear forces and tensile force, perpendicular to the shear direction. We hoped that a combined implementation of volume preservation and tensile strain resistance would lead to a more accurate shear strain resistance. Figure 5.2 shows graphs for the tensile force and the shear force. Looking at the graph for shear force with $\nu = 0$ already shows that our attempts are a bit of a long shot. Without volume preservation, the simulated shear resistance is far below the expected value. Increasing Poisson's ratio would normally lower the expected shear resistance but should relatively increase the simulated resistance for the test to be successful. What we see is that with $\nu = 0.3$, the simulated resistance does actually increase a little in the first part of the plot, so the simulated values and expected values have approached each other a bit. Perpendicular to the shear there should be no force at all, but figure 5.2(a) show that is still a considerable force present in that direction. Enabling volume preservation does reduce this force, but only very little.

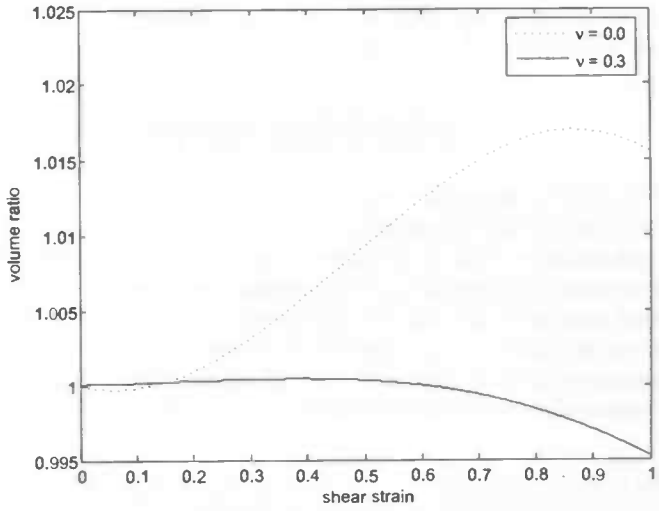
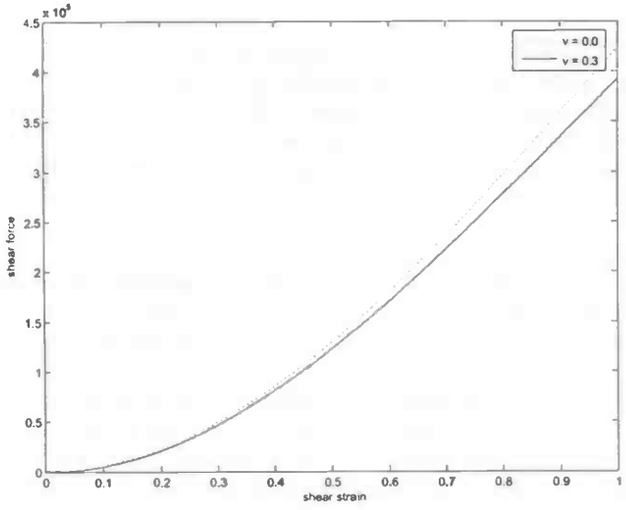


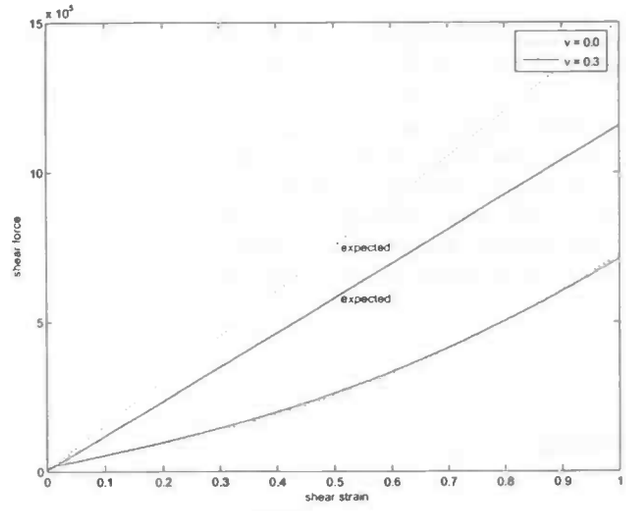
Figure 5.1: Volume ratio

5.3 Evaluation

The goal was to control shear resistance by adjusting Poisson's ratio. What our figures show for the approach discussed in this chapter is that despite volume is preserved for higher values of Poisson's ratio, it hardly affects the resisting forces. That for $\nu = 0.3$ the shear resistance plot is closer to the expected value can for the larger part be explained by a decreasing expected resistance. The simulated shear resistance increases a little for higher Poisson's ratio, but if the model would be in consensus with continuum mechanics theory the simulated shear resistance should have dropped as well. Pure shear like we examined normally does not result in any volume change. That different amounts of volume preservation have no significant effect on shear resistance makes sense in that context.



(a) Tensile force



(b) Shear force

Figure 5.2: Shear and tensile strain

6 Three dimensional springs

In the last chapter the CMPM method was discussed and the attempts to loosen the requirements of the methods. During this search my supervisor stumbled upon something called the Virial theorem. In short this theorem can be used to find the pressure or stress within a finite system of point particles which are interacting with each other. As it turned out the virial theorem can also be used to find the forces of the corner points of a tetrahedron under some specified stress. Later we found another method to calculate forces on corner points of the tetrahedron when the stress inside the volume of the tetrahedron is known. This last method is based on basic stress theory.

Where the mass-spring model is one-dimensional, this model is essentially a three-dimensional spring model. Were we working mostly with strain and stresses expressed in scalars, now these are replaced by tensors. Tensor based continuum mechanics as described in chapter 2 can directly be applied in this model.

In this chapter we will go into some deformable media simulation models which are like the CMPM based on a tetrahedralized volume. The state of the volume is determined by the positions and velocities of the vertices of the tetrahedra. At the position of each vertex we place a particle, that is assigned a quarter of the mass of each neighbouring tetrahedron. The vertices will move along with their associated particles. Based on the stress in each tetrahedron due to deformation, forces are calculated for the particles at each time step. With these forces the status is updated by numerical integration.

The first section explains how to find the stress in a tetrahedron based on the vertex positions in undeformed state and the vertex positions in a deformed state. Uniform displacement and rotation are both eliminated and the result is the stress tensor orientated for the tetrahedron in the deformed state. In the following sections some methods to find forces for the tetrahedron vertices based on its stress tensor. First we discuss a method, where the tensor is used to calculate the forces on the faces of the tetrahedron and distributed over the vertices. Second we demonstrate how the forces acting on the vertices can also be found using the virial theorem. The forces acting upon the particles are the sum of the forces found by processing all neighbouring tetrahedra. In the end of this chapter the virial theorem method for calculating forces will be put through some simple tests to give a basic impression on how it performs on processing speed, force approximation quality, stability and volume preservation.

6.1 Tetrahedral stress

The four vertices of the tetrahedron describe its shape and its displacement. Their positions must be non-coplanar, which would otherwise reduce the volume of the tetrahedron to zero. We are only interested in the deformation and not in their displacement. To rule out the translation we express the shape in three points in relation to the fourth point (figure 6.1). Later we will also discuss the possible rotation of the tetrahedron.

The relation between two differently shaped tetrahedra is a linear transformation of the positions of its vertices. Consider the same tetrahedron in two different shapes. The shape of the undeformed tetrahedron is described by the vertices $\mathbf{p}_{1..3}$ and the shape of the deformed tetrahedron by vertices $\mathbf{q}_{1..3}$. Each of the vertices i in the undeformed tetrahedron $\mathbf{p}_i$ corresponds with the deformed tetrahedron $\mathbf{q}_i$. Each of these three vectors are handled as if they were base vectors of a skew coordinate system. By placing the base vectors inside a 3×3 matrix we have a description the shape and orientation of the coordinate system. In (6.1) and (6.2) is demonstrated how matrix A for the undeformed tetrahedron and the matrix B for the deformed tetrahedron

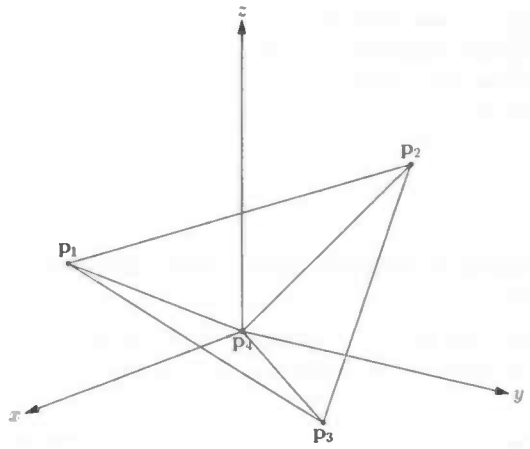


Figure 6.1: The shape of the tetrahedron expressed in three vertices

are composed.

$$\mathbf{A} = \begin{pmatrix} p_{1x} & p_{2x} & p_{3x} \\ p_{1y} & p_{2y} & p_{3y} \\ p_{1z} & p_{2z} & p_{3z} \end{pmatrix} \tag{6.1}$$

$$\mathbf{B} = \begin{pmatrix} q_{1x} & q_{2x} & q_{3x} \\ q_{1y} & q_{2y} & q_{3y} \\ q_{1z} & q_{2z} & q_{3z} \end{pmatrix} \tag{6.2}$$

A linear transformation exists which transforms system A into system B (6.3). This transformation is likely to contain some rotation matrix **R** that matches the orientation of A to that of B. Another matrix **U** deforms A so that it is matched entirely to B.

As we see in equation (6.3), system A is first orientated similar to B by a rotation matrix R. The consequence for the deformation matrix U is that its orientation matches that of B. Later we need to find forces to counteract the deformation of B. Since these forces must also match the orientation of B, the deformation described in U suits our needs as a deformation matrix to base a stress tensor on. First we isolate the combined transformation of rotation **R** and deformation **U** by moving A to the left side of the equation (6.4).

$$\mathbf{B} = \mathbf{URA} \tag{6.3}$$

$$\mathbf{BA}^{-1} = \mathbf{UR} \tag{6.4}$$

We have determined that **U** is already orientated the way we like, so we only need to make the rotation matrix **R** disappear. Multiplying the linear transformation by its transpose does the trick (6.5). The transpose of a rotation matrix is equal to its inverse. Placing **R** next to its transposed version $\mathbf{R^T}$ eliminates both **R** and $\mathbf{R^T}$. Note that this is the first step in a procedure called polar decomposition. What this procedure does is decomposing a matrix in a orthonormal or rotation matrix and a symmetric matrix. The second step would involve finding the orthonormal matrix **R**, by taking the square root of the matrix $\mathbf{U}^2$

$$\mathbf{UR(UR)^T} = \mathbf{URR^TU^T} = \mathbf{UIU^T} = \mathbf{U}^2. \tag{6.5}$$

Still **U** is not yet found, only a squared version $\mathbf{U}^2$. Calculating the square root of the matrix involves principal component analysis which, if performed for every tetrahedron in every iteration, would be a major performance limiting operation. We therefore assume that the deformation is small so that the terms e in (6.6) are also small. The matrix **U** is now close the identity matrix. Only using the linear part of $\mathbf{U}^2$ will not be very different from the real value of **U**.

$$\mathbf{U} = \begin{pmatrix} 1+\varepsilon_{xx} & \varepsilon_{xy} & \varepsilon_{xz} \\ \varepsilon_{xy} & 1+\varepsilon_{yy} & \varepsilon_{yz} \\ \varepsilon_{xz} & \varepsilon_{yz} & 1+\varepsilon_{zz} \end{pmatrix} \tag{6.6}$$

After subtracting the identity matrix from $\mathbf{U}$ (6.7) we have a reasonable approximation for the strain tensor ε. Now that we have the strain, the stress is calculated using the formulas in 2.36. The stress tensor found by this method does not only describe the stress for one point, but is assumed to be uniform for the volume of the tetrahedron.

$$\varepsilon \approx \frac{1}{2}(\mathbf{U}^2 - \mathbf{I}) \tag{6.7}$$

6.2 Calculating forces I: Tetrahedron face method

In the previous section we uncovered an in our opinion acceptable approximation for the stress tensor over the volume of a tetrahedron. In this and the following section this stress tensor is used for finding forces counteracting the strain.

Here we will use some stress theory as it was discussed in section 2.2. First we focus our attention on a single tetrahedron and from there we proceed to the forces on the particles in the volume. With a stress tensor the stress for a face in any direction can be calculated. The hull of a tetrahedron consists of four triangular faces. We can calculate the normalized direction $\hat{\mathbf{n}}$ of these faces and their surface area a with the positions of the tetrahedrons vertices. Multiplying the direction $\hat{\mathbf{n}}$ with the stress tensor σ and multiplying the resulting stress with the area of the face a leads to the force $\mathbf{f}$ acting on the surface of the face (6.8). Note that when the direction $\hat{\mathbf{n}}$ of a triangular face is calculated by the cross-product, the length of the resulting vector is twice the face area. If this vector is multiplied by the stress tensor σ, the result must be divided by two to get the right amount of force $\mathbf{f}$. Equation (6.8) demonstrates how this works for a face determined by vertices $\mathbf{p}_{1..3}$.

$$\mathbf{f} = \sigma\hat{\mathbf{n}}a\mathbf{f} = \frac{1}{2}\sigma((\mathbf{p}_2 - \mathbf{p}_1) \times (\mathbf{p}_3 - \mathbf{p}_1)) \tag{6.8}$$

The integration method we used was based on point masses, so the forces for the faces must somehow be distributed to the vertices of the tetrahedron. A simple way of doing this is to assign a third of the force on a face to each of its three vertices. Doing this for all faces of the tetrahedron, every vertex is assigned a third of the forces from three faces. Effectively this is the same as assigning a third of the force from the face opposite to the vertex in the tetrahedron. The sum of the face normals $\hat{\mathbf{n}}_{1..4}$ times the surface areas $a_{1..4}$ for the faces of a tetrahedron is zero (6.9). This is still true when the linear force tensor is applied for each face (6.10). When one of the faces is moved to the other side of the equation (6.11) one can easily see that a third of the forces of three faces 1..3 combined is equal to a third of the reversed force of face 4.

$$|\hat{\mathbf{n}}_1 a_1 + \hat{\mathbf{n}}_2 a_2 + \hat{\mathbf{n}}_3 a_3 + \hat{\mathbf{n}}_4 a_4| = 0 \tag{6.9}$$

$$|\sigma\hat{\mathbf{n}}_1 a_1 + \sigma\hat{\mathbf{n}}_2 a_2 + \sigma\hat{\mathbf{n}}_3 a_3 + \sigma\hat{\mathbf{n}}_4 a_4| = 0 \tag{6.10}$$

$$\sigma\hat{\mathbf{n}}_1 a_1 + \sigma\hat{\mathbf{n}}_2 a_2 + \sigma\hat{\mathbf{n}}_3 a_3 = -\sigma\hat{\mathbf{n}}_4 a_4 \tag{6.11}$$

Now that the procedure for calculating the forces of the vertices on a single tetrahedron is established, the forces on the particles can be found quite easily. Each particle is associated with the vertex for one or more tetrahedra. The force for each particle is the sum of the forces it gets from all neighbouring tetrahedra. Finally the direction of the force must be reversed so that it no longer represents the forces causing the stress but rather the material's response to the stress.

6.3 Calculating forces II: Virial Theorem method

The virial theorem is a law that is well known in fields like astronomy and gas theory. It states that for a stable, self-gravitating bound distribution of point masses the time average kinetic energy K can be found by equation 6.12. It was originally proposed by Rudolf Clausius who did during his lifetime important work in thermodynamics. When the kinetic energy is obtained, finding the pressure tensor is only one easy step away.

$$\mathbf{K} = \frac{1}{2}\sum(\mathbf{x}_i\mathbf{f}_i^{\mathrm{T}}) \tag{6.12}$$

(6.13)

It is not hard to understand the virial theorem. There are several ways to derive (6.12) and what we present here is just one of them. An important criterion is that the position of the masses in the system is bounded. Formally a function g is bounded if equation 6.14 holds for all t_1 and t_2 and some constant k. No particles will cross a certain spatial boundary at any time during their trajectory. This implies that over a long period of time, the average position $\mathbf{x}$ does not change and the average velocity $\dot{\mathbf{x}}$ goes to zero.

$$|g(t_2) - g(t_1)| < k \tag{6.14}$$

When both position $\mathbf{x}$ and velocity $\mathbf{v}$ are bounded over time, their product is also bounded over time (6.15). Therefore the average time derivative of the product of position and velocity converges to zero when the average is taken over a sufficiently large amount of time.

$$\frac{d(\mathbf{x} \cdot \mathbf{v})}{dt} = \mathbf{x} \cdot \mathbf{v} + \mathbf{v}^2 = 0 \tag{6.15}$$

The value of $m\ddot{r}$ equals the net force on the particle and $m\dot{r}^2$ twice the kinetic energy. Thus, when both sides of equation 6.15 are multiplied by m and subsequently solved for the kinetic energy $\mathbf{K}$ we find Clausius' virial theorem proposition as shown in equation 6.12.

The virial theorem gives the time average kinetic energy $\mathbf{K}$ in tensor form, based on the position and net forces exerted on the particles by other particles. Pressure can be considered as the kinetic energy per volume, thus by dividing the kinetic energy by the bounding volume the particles reside within, the pressure tensor is found (6.16), which is equal to the stress tensor.

$$\sigma = \frac{1}{2V} \sum (\mathbf{x}_i \mathbf{f}_i^{\mathbf{T}}) \tag{6.16}$$

In our case the stress is known for every tetrahedron by the method described in the previous section. For $\mathbf{x}_i$ we use the vertices from the tetrahedron. The forces $\mathbf{f}_i$ acting on these vertices are the only unknowns in the equation. Finding the forces can be done by solving a 9×9 system (6.17). The structure of the matrix suggests a more efficient method, namely solving a 3×3 system for three different vectors (6.18).

$$\frac{1}{2V}\begin{pmatrix} q_{1x} & q_{2x} & q_{3x} & 0 & 0 & 0 & 0 & 0 & 0 \\ q_{1y} & q_{2y} & q_{3y} & 0 & 0 & 0 & 0 & 0 & 0 \\ q_{1z} & q_{2z} & q_{3z} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & q_{1x} & q_{2x} & q_{3x} & 0 & 0 & 0 \\ 0 & 0 & 0 & q_{1y} & q_{2y} & q_{3y} & 0 & 0 & 0 \\ 0 & 0 & 0 & q_{1z} & q_{2z} & q_{3z} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & q_{1x} & q_{2x} & q_{3x} \\ 0 & 0 & 0 & 0 & 0 & 0 & q_{1y} & q_{2y} & q_{3y} \\ 0 & 0 & 0 & 0 & 0 & 0 & q_{1z} & q_{2z} & q_{3z} \end{pmatrix} \begin{pmatrix} f_{1x} \\ f_{2x} \\ f_{3x} \\ f_{1y} \\ f_{2y} \\ f_{3y} \\ f_{1z} \\ f_{2z} \\ f_{3z} \end{pmatrix} = \begin{pmatrix} \sigma_{11} \\ \sigma_{21} \\ \sigma_{31} \\ \sigma_{12} \\ \sigma_{22} \\ \sigma_{32} \\ \sigma_{13} \\ \sigma_{23} \\ \sigma_{33} \end{pmatrix} \tag{6.17}$$

$$\begin{aligned} \frac{1}{2V}\begin{pmatrix} q_{1x} & q_{2x} & q_{3x} \\ q_{1y} & q_{2y} & q_{3y} \\ q_{1z} & q_{2z} & q_{3z} \end{pmatrix} \begin{pmatrix} f_{1x} \\ f_{2x} \\ f_{3x} \end{pmatrix} &= \begin{pmatrix} \sigma_{11} \\ \sigma_{21} \\ \sigma_{31} \end{pmatrix} \\ \frac{1}{2V}\begin{pmatrix} q_{1x} & q_{2x} & q_{3x} \\ q_{1y} & q_{2y} & q_{3y} \\ q_{1z} & q_{2z} & q_{3z} \end{pmatrix} \begin{pmatrix} f_{1y} \\ f_{2y} \\ f_{3y} \end{pmatrix} &= \begin{pmatrix} \sigma_{12} \\ \sigma_{22} \\ \sigma_{32} \end{pmatrix} \\ \frac{1}{2V}\begin{pmatrix} q_{1x} & q_{2x} & q_{3x} \\ q_{1y} & q_{2y} & q_{3y} \\ q_{1z} & q_{2z} & q_{3z} \end{pmatrix} \begin{pmatrix} f_{1x} \\ f_{2x} \\ f_{3x} \end{pmatrix} &= \begin{pmatrix} \sigma_{13} \\ \sigma_{23} \\ \sigma_{33} \end{pmatrix} \end{aligned} \tag{6.18}$$

When only one 3×3 system needs to be solved, LR decomposition with backward substitution might be a good choice. Because the matrix in the three systems (6.18) are all the same, a computationally efficient method for solving for $\mathbf{f}$ would be to multiply the appropriate elements from σ with the inverted matrix. For a 3×3 matrix its inverse can be calculated in a non-iterative way and the result can be used for three vectors.

6.4 Damping

We first talked about damping in section 3.4, where kinetic energy is lost due to internal friction. For this model damping seemed necessary to maintain stability. What we noticed when simulating an undamped material was that kinetic energy seemed to slowly accumulate, originating from parts that had been under relatively large stress and spreading through the entire model. Very little damping is enough to stop the energy from increasing. Higher damping for instance reduces the effects of collisions, because much of the kinetic energy is absorbed.

The damping for the mass-spring model that was described there is still very usable here. We have thought about a three dimensional damping model where the difference in velocity between the four particles of a tetrahedron and the average velocity of the four particles together was damped. A disadvantage was that rotation was damped as well, so that under a strong damping constant b, the whole body would stop rotating entirely when there were no outside influences.

So we fell back on the one dimensional mass-spring damping, which is based on leakage in the velocity difference between particles that form an edge together. In section 3.4 is described how this works.

6.5 Numerical integration

If possible we would like to use the implicit integration scheme for this model, because of its stable characteristics. To implement the implicit integration we need to have the force gradients at every time step. We were able to calculate those for the mass spring model and the CMPM, but here the forces are found in an entirely different way. For both the tetrahedron face method and the virial theorem method it is very difficult and expensive to find these gradients. In [MKN+] they faced similar problems and eventually came to the conclusion that it does not pay off to use the implicit integrator over an explicit integration method like leapfrog integration. The larger time steps they could take with the implicit integrator were compensated by the calculation of the force gradients. Therefore we decided to use just leapfrog integration as well.

6.6 Simulation results

In previous sections we have discussed two methods to calculate forces on particles to counteract the deformation of a tetrahedron. In this section they will be put to the test to see how the methods of tetrahedral stress perform in general. Some interesting results found during simulations are presented. These results are concerned the computing speed, the amount of force in response to different kinds of transformations over a body and volume preservation.

The boundary mesh input for the simulation program was modeled by hand in 3D studio MAX. After it the mesh was loaded, a tool called tetgen [Si] was used to tetrahedrise the volume described by the mesh.

We start with an indication under what circumstances the model remains stable. It is important for a stable simulation that the tetrahedrisation is of a certain amount of quality in the sense that the angles in a tetrahedron are not too small, otherwise, the edges in the model may differ substantially. When the particle of a large edge moves some amount towards or away from the other particle the deformation may be small, but a displacement of the same order of magnitude involving a small edge may be a very large deformation. Large deformations lead to strong forces, especially for materials with a high Poisson's ratio. Strong forces lead to large accelerations and that is what makes a model unstable. What also influences the capability of a body to handle large deformations is its size. When the size of a body increases by a factor s in all directions, the mass of the particles grows cubically with s^3. The strength of the forces resulting from a similar deformation will be the same for a large and a small body, but not the acceleration since that also depends on the mass. If the size of a model is increased by a factor two in all directions, the particle accelerations will be eight times smaller. Obviously the acceleration can also be controlled with the density parameter, to some extent.

With a density of $1000\ \frac{kg}{m^3}$ and normal earth gravity we have tested the stability for a solid sphere. This sphere was composed of 381 particles, 955 tetrahedra and had a volume of approximately $4m^3$. We took 700 time

integration steps per simulated second. The sphere was dropped from a height of 7 meters and bounced off the floor plane. A damping constant of $100\frac{N}{ms^{-1}}$ kept vibrations in the material under control. With Poisson's ratio set to 0.1 the simulation was stable when Young's modulus was set somewhere in between 2×10^5 and 3×10^6. A lower value would make the sphere collapse on itself on impact and a higher value led to instability resulting in an 'explosion'. Setting Poisson's ratio to a higher value resulted in reduced tolerance for both low and high values for Young's modulus. For the a ratio of 0.4, a value for Young's modulus between 4×10^5 and 1×10^6 gave no problems.

Running the sphere simulation on a AMD XP 1800+ processor we were able to approximate real-time by approximately 45%. The graphics were rendered on a Radeon 9800 Pro graphics card. The visuals were refreshed at a rate of $40hz$ and hardly affecting the simulation performance.

We did the bounce simulation with a more complex body. Figure 6.2 shows a sequence of snapshots at the moment the object hits the floor. During the collision, the bars by which the model is composed bend to absorb the collision forces. The simulation showed well the residual vibrations going through the body after it had bounced up and had lost contact with the floor.

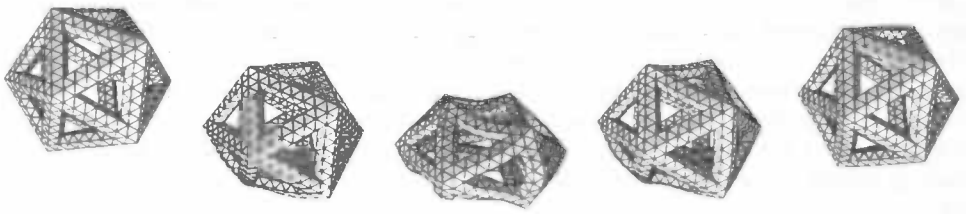


Figure 6.2: Complex object bouncing of the floor

During strong collisions the particles in a tetrahedron sometimes move through its opposite plane, making the tetrahedron flip. The model's response to this event is interesting and might be considered either a bug or a feature. The model recognizes the tetrahedron flip as a form of rotation so that the flip is not undone but treated is if it was not flipped at all. For flipped tetrahedron the strain starts working in the other direction, making that the tetrahedron remains flipped unless some action from outside the tetrahedron 're-flips' it. The nice thing about this may be that under strong forces the model forms dents in the most pressured region of the body and stability is preserved. Unless the whole body turns inside out, the simulation does not run out of control.

Mass-spring models or the CMPM outperform this model easily in terms of computing speed. The large amount of linear algebra that has to be done for every simulation step for every tetrahedron is responsible for the difference. If it is not for performance one would choose this model, then it should give some advantage in terms of quality. A material that approaches the real world more accurately, a more stable simulation or more capabilities is what would keep this model interesting.

To see how realistic the material behaves, we used a cylinder shape and ran some tests on it. Assuming the cylinder should behave as the theory from section 2.3 we used that as a reference to compare our results to. The cylinder was 4 meters long, had a diameter of 1.5 meters and composed of 221 particles and 492 tetrahedra. We set Young's modulus at 1×10^6, Poisson's ratio at 0.4. Damping was not really relevant here. Figure 6.6 shows how volume is preserved during stretching. Volume preservation is eliminated when Poisson's ratio is set to 0. A negative Poisson's ratio is also supported as figure 6.3(b) demonstrates. The longer the cylinder is stretched, the wider it becomes.

We measured how strong a body would resist against tensile and shear stress and if the magnitude of these force would be close to the expected magnitude based on the material parameters. Our test body was again a cylinder but a shorter one. When shearing the long cylinder there is clearly no pure shear as we saw the cylinder bend at both sides. Though this would probably happen just the same in the real world we considered it to be undesirable when measuring shear forces. The bending was less profound with a shorter type of cylinder. The dimensions of the cylinder were a length of 0.5 meters and a diameter of 2 meters. Young's modulus was set to 1×10^6 and Poisson's ratio to 0.2. For these tests damping was irrelevant.

(a) Cylinder stretching with $\nu = 0.4$

(b) Cylinder stretching with $\nu = -0.4$

The particles at the top and at the bottom were forced to move away from each other. Figure 6.4(a) shows how strong the simulated material resisted stretching the cylinder. The combined force acting on the top particles are plotted. Additional 'expected' plots show how the material would react if it was purely linear, independent of the amount of deformation. As one can see the material's response is close to expected for small strains and starts to deviate from the expected line for larger strains in the first stages. The tensile force plot curves away from the expected line. At zero deformation the tensile force gradient seems to match the expected coëfficient closely. Other tests with varying Poisson's ratio showed that the higher Poisson's ratio, the higher the initial tensile force gradient and the more the plot diverges from the expected plot. For a Poisson's ratio of 0, the two plots lined up exactly under small strain. Shearing forces do not show up, as one would expect for tensile deformations.

Next the particles on the top and the bottom were forced to move away from each other sideways to provoke shear strain. Figure 6.4(b) shows that the shear force plot is not curved as the tensile force plot was, but resembles much more a straight line. The gradient however doesn't seem to match the coefficient of the expected line, still the shear force is pretty close to the expected shear force. The tensile force starts with a zero gradient as it should, however it does not stay there as it becomes stronger under more shear. A larger Poisson's ratio lowers the shear force like it should, unlike the model in chapter 5. On the other hand, a larger Poisson's ratio also strengthens the tensile forces by some factor.

Tests with different values for Young's modulus showed that the measured tensile force and measured shear force is proportional to Young's modulus.

This discussion may seem a bit vague and probably the outcome depends heavily on the test body. The point of this all is to provide a rough sketch of how the model behaves for a simple model and how the forces may change when the material parameters change.

Finally we experimented a bit with homogeneity. A body with varying density was dropped onto the floor. The results are shown in figure 6.3. The heaviest side is compressed most when the body touches the floor. The heaviest part also the side bounces the latest and least far from the ground.

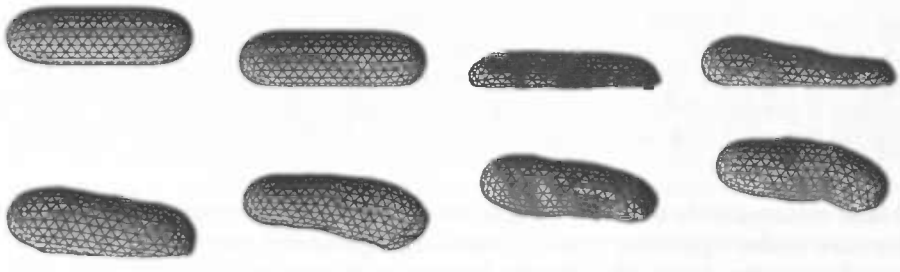
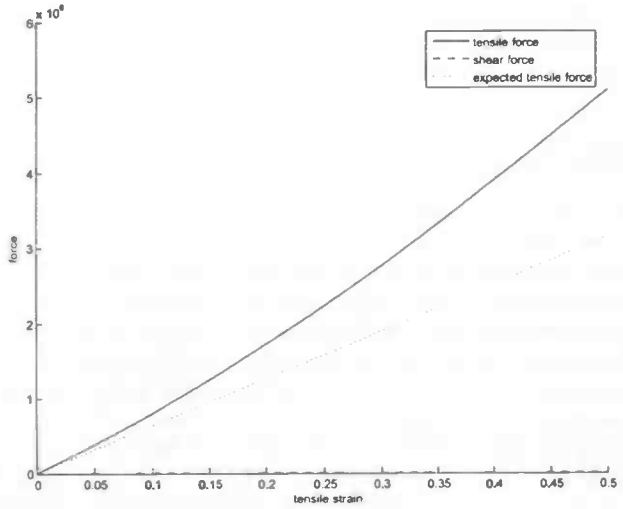
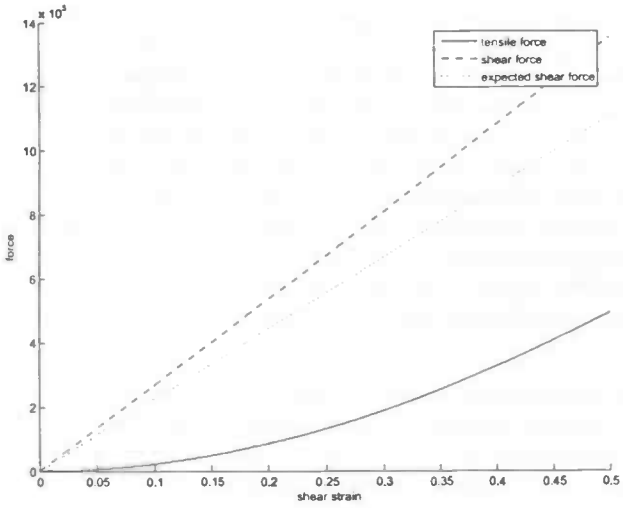


Figure 6.3: Inhomogeneity by gradually changing density. The right side is the heaviest.



(a) Tensile force



(b) Shear force

Figure 6.4: Shear and tensile strain

6.7 Evaluation

In this chapter we introduced a new type of interaction between point masses. The new type of spring is not a one dimensional spring between two point but a three dimensional spring between four points. The theory from continuum mechanics can directly be applied to get a very decent response to deformation. We demonstrated how to find the strain tensor from a deformed tetrahedron by comparing it to the tetrahedron in undeformed state. The stress tensor which could be found directly from the strain forms the basis for deformation resisting forces on the particles. Two methods two calculate these forces were presented, which are both pretty much similar in terms of results and processing cost. Some results were given for one method based on the virial theorem. Though the simulation rate can not compete with the one dimensional mass-spring models, the material seems to mimic the real materials surprisingly well for an un-tweaked model. We saw that the material could be controlled by adjusting the material parameters. Shear and tensile resistance and the amount of volume preservation all responded to changing values for Young's modulus and Poisson's ratio, though the results did not match up exactly with our expectations. Whether the model is wrong or our expectations remains to be seen.

7 Conclusion

In this conclusion we will first summarize in few words what this paper has been about. After introducing preliminary concepts on continuum mechanics and numerical integration, three different methods were examined. The first one CMPM, originally described in Everts' paper on deformable objects was dissected and some critical flaws in the model were uncovered. In the same chapter there is also a discussion about attempts to simplify the CMPM by eliminating the Voronoi graph. This we would probably not have been bothered with if we had found out earlier about the model's incorrectness. The following chapter is about an experiment based on the relation between Young's modulus, Poisson's ratio and the shear modulus. We hoped to get an answer to the question whether shear could indirectly be controlled by actively implementing tensile strain handling and volume preservation. In chapter 6 we described a model where we experiment with a model that has a new type of interacting medium between its point masses.

For the CMPM described in chapter 4 the conclusion is that in its current state, it is not clear what the model's advantage is over a mass spring model. The strain decomposition allows users to set different constants for different types of strain. The components in which the stain is decomposed can not righteous be classified as tensile strain or shear strain so that the effect of the material parameters in the model is not straightforward and intuitive. When a good reason is found why the strain decomposition has an advantage over other force calculation methods this model might prove valuable after all, but until then it remains an overly complex and computational cost expensive mass-spring implementation. In general one might add that a method with local statically directed constraint components (like springs) is perhaps not a good idea to construct an isotropic model where distinction is made between tensile forces and shearing forces. Directed constraint components will by definition not behave the same in different orientations, under similar circumstances. Tensile and shear strain will not be recognized correctly and unless the interaction network compensates for this, the result will be anisotropic behaviour.

Can shear be controlled by implementing a combination of tensile strain handling and volume preservation? The answer we must give is that our implementation described in chapter 5 does not do a very good job. In our model, which is probably too simplistic, we were unable to close in on the mechanism that makes that the three constants to be related to each other. A mass-spring model combined with a naive type of volume preservation does hardly change shear resistance, not in the least part because pure shear does normally not change the volume of a body. Though the volume preservation technique was very simple, volume was preserved much better than without indeed, so on that part it was a success.

If any type of model in this paper deserves another closer look than in our opinion it is the one described in chapter 6. No longer are we relying on one dimensional springs for interaction, but instead we have a three dimensional spring in the shape of a tetrahedron. The advantage over a one-dimensional spring is that three-dimensional deformations are recognized and handled. A disadvantage may be that we are no longer dealing with scalars, but with three-dimensional tensors which are computationally more expensive and for some more difficult to understand. The simulation results in our implementations are the best in this paper in terms of realistic material behaviour, but at the cost that the simulation rates are lower than those for mass-spring models. Another reason why the model is slower than mass-spring is that the fast and stable implicit integration is no real option since calculating the force gradients is too expensive. The technique of finding the stress inside a tetrahedron by a linear transformation and then solve the virial theorem for forces seems to be original since we were unable to find any written documentation about similar experiments. It is also not very hard to implement. Virial theorem may not be known in the field of deformable models as such, but a similar technique is used to determine energy of deformation for expressing equilibrium equations in FEM. Using continuum mechanics to move from strain to stress should lead to accurate stress, but especially the theory on strain specifies that for small deformations results are reasonably accurate. In real-time applications like virtual surgery, organic tissue allows strain of over 100%. The model does allow for such deformations though they might be less close to reality. We use a linear approximation for the a squared strain matrix which introduces an extra truncation error.

Some issues are still open. One of them is the intolerance of the model toward low or even moderate quality tetrahedrisations. Small angles in tetrahedra quickly lead to numerical instability. Not only must one put extra effort in providing quality volumetric body input, it also raises additional obstacles for permanent mutilation like cutting. Cutting through tetrahedra may cause very small angles. On the other hand, when tetrahedra are left intact a simulated object could very easily be cut by splitting particles. Particles are the only bond between the tetrahedra and assigning a new particle to a severed tetrahedron should not be difficult.

Future work should depend on the application requirements. When improved realism is required, more tests on the range of materials that are currently supported and how well reality is approximated are probably worth the effort. Based on the approximation quality the model may be tuned or calibrated, though we think it is not very likely that for linear materials large improvements can be achieved without complicating the design. One may look into expanding the range of supported materials with anisotropy, inhomogeneity and non-linearity. The model should be able to support anisotropic materials when also the asymmetric orientation tensor is extracted besides the deformation tensor.

When high simulation rates are most important, such as in most computer games, we think that three dimensional springs are probably not the answer. When one is willing to pay the processing price for a more natural acting material, this type of model might be worth considering.

Bibliography

[BF05] Richard L. Burden and J. Douglas Faires. *Numerical Analysis*. Thomson Brooks/Cole, eigth edition, 2005.

[BW98] David Baraff and Andrew Witkin. Large steps in cloth simulation. *Computer Graphics*, 32(Annual Conference Series):43–54, 1998.

[CE88a] B. D. Caddock and K. E. Evans. Mircoporous materials with negative poisson's ratios: I. mechanisms and interpretations. *Applied Physics*, 137(1.2), 1988.

[CE88b] B. D. Caddock and K. E. Evans. Mircoporous materials with negative poisson's ratios: I. microstructure and mechanical properties. *Applied Physics*, 137(1.2), 1988.

[dBvKOS97] Mark de Berg, Marc van Kreveld, Mark Overmars, and Otfried Schwarzkopf. *Computational geometry: algorithms and applications*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, second edition, 1997.

[Eva88] K. E. Evans. Tensile network microstructures exhibiting negative poisson's ratios. *Applied Physics*, 137(1.2), 1988.

[Eve06] Maarten Everts. Simulation of deformable objects, 2006. Supervisor H. Bekker.

[FB68] M. Filonenko-Borodich. *Theory of Elasticity*. P. Noordhoff N.V., 1968.

[GM97] S. Gibson and B. Mirtich. A survey of deformable modeling in computer graphics, 1997.

[Hau04] Michael Hauth. *Visual Simulation of Deformable Models*. PhD thesis, Wilhelm-Schickard-Institut für Informatik, University of Tübingen, Germany, July 2004.

[Hei01] J. H. Heinbockel. *Introduction to Tensor Calculus and Continuum Mechanics*. Trafford Publishing, 2001.

[JP99] Doug L. James and Dinesh K. Pai. Artdefo - accurate real time deformable objects. In Alyn Rockwood, editor, *Siggraph 1999, Computer Graphics Proceedings*, pages 65–72, Los Angeles, 1999. Addison Wesley Longman.

[MKN+] M. Müller, R. Keiser, A. Nealen, M. Pauly, M. Gross, and M. Alexa. Point based animation of elastic, plastic and melting objects.

[Si] Hang Si. Tetgen website, `http://tetgen.berlios.de/`.

[Woj88] K. W. Wojciechowski. Two-dimensional isotropic system with a negative poisson ratio. *Applied Physics*, 137(1.2), 1988.