Using CONTACT in dynamical simulations

Interim thesis report at VORtech
by Hugo de Looij

as part of the program Applied Mathematics at the
Delft University of Technology

Supervisor (VORtech)	Dr. ir. E.A.H. Vollebregt
Supervisor (TUD)	Prof. dr. ir. C. Vuik

April 9, 2015

Contents

1 Introduction

Contact mechanics is the theory that deals with the deformation of contacting objects. It is an import research topic for many different industries, in particular for the rail industry. As an example, consider a train moving over the tracks. Because of gravity, the train will exert a large force on the rails. This results in deformation of both the wheels and the rails. It is important to understand this process, so that the rail industry can estimate and prevent the possibility of rail deformation, estimate the wear and tear of the rails and wheels, as even estimate the probability of train derailment. Contact mechanics, or in this case frictional contact mechanics, gives us the tools to understand this process.

As the name would suggest, the CONTACT software solves contact problems between two objects. It has originally been developed by Joost Kalker. In 2000, VORtech has taken over the software. It is now being further developed by Edwin Vollebregt, who is my supervisor for this Master project.

The software can be used for a large amount of (homogeneous) contact problems and can be used to compute deformations, determine the forces that are being exerted on the surface, and determine on which areas of the contact surface slip will occur. It aims to be the worlds fastest detailed contact model.

In this literature report, we will describe the physics and mathematics of dynamical contact problems, i.e. time-dependent contact problems. We will describe the basics of elasticity theory, contact mechanics (Hertz theory in specific) and time integration schemes for dynamical problems. To get a better understanding of this theory, we will apply this for two simple dynamical contact problems.

The main research goal for the project following the literature study is to understand how CONTACT can be used in combination with a sufficient time integration scheme for dynamical contact problems. This can be fairly complex if we allow both local and global deformations and/or friction to occur. To validate whether the CONTACT approach is accurate, we would also like to solve the same problems using a Finite Element approach.

2 Introduction to contact mechanics & elasticity theory

Contact mechanics is the study that deals with the physics behind the contact of two bodies. It involves the computation of the pressure, contact forces, and deformation in either static or dynamic problems. The bodies might have different elastic properties which can result in a different distribution for the pressure and a different deformation of the objects.

As an example, a train will exert a large force on the rails. This results in deformation of the wheels and the rails. The magnitude of this deformation, however, depends on properties such as the material of both objects and the geometry of both objects around the initial contact point.

Note that this chapter is explained for the three dimensional case. The two dimensional case is similar. We use the Cartesian coordinate system and a point $\mathbf{u}$ can be denoted both as $\mathbf{u} = (u_x, u_y, u_z)$ as well as $\mathbf{u} = (u_1, u_2, u_3)$.

2.1 Displacements

Consider an undeformed object in rest and take one particle of this object. Now consider a force at the boundary of the object pushing in any direction. This can trigger the particle to move slightly, i.e. the object deforms. This translation of the particle is called the displacement. If this displacement is zero, that means there is no deformation at this particle. There are three displacement variables, one for each direction: u_x, u_y, and u_z.

2.2 Strain

Strain represents the stretching (or indenting) of an object. Once again, take one particle of this object in its undeformed state. Take another particle close to this one. If the body is deformed, then the distance between the two particles can differ. Strain is the relative change of the position of points in the body and is therefore a dimensionless quantity.

There are two different kinds of strain, namely the longitudinal strains $\varepsilon_{xx}, \varepsilon_{yy}$, and ε_{zz}, as well as the shearing strains $\varepsilon_{xy}, \varepsilon_{xz}, \varepsilon_{yx}, \varepsilon_{yz}, \varepsilon_{zx}$, and finally ε_{zy}. The longitudinal strains corresponds to the relative change of the position of points in the body in the corresponding direction. If a homogeneous bar of $1m$ width is uniformly stretched to $1.5m$, then the strain $\varepsilon_{xx} = \frac{3}{2}$. The shear strains represent the change of angle between two points as their distance tends to zero. These shear strains are always symmetric, so for example $\gamma_{xy} = \gamma_{yx}$.

All nine strain components can be added together to create the so-called strain tensor, which is defined by

$$\varepsilon = \begin{pmatrix} \varepsilon_{xx} & \gamma_{xy} & \gamma_{xz} \\ \gamma_{yx} & \varepsilon_{yy} & \gamma_{yz} \\ \gamma_{zx} & \gamma_{zy} & \varepsilon_{zz} \end{pmatrix} \tag{2.1}$$

2.3 Strain-displacement relations

Note that the longitudinal strains are simply the derivative of the displacement in the same direction. The strain components are therefore connected to the displacements by the following equations

$$\begin{aligned} \varepsilon_x &= \frac{\partial u_x}{\partial x} \\ \varepsilon_y &= \frac{\partial u_y}{\partial y} \\ \varepsilon_z &= \frac{\partial u_z}{\partial z} \\ \gamma_{xy} &= \frac{\partial u_x}{\partial y} + \frac{\partial u_y}{\partial x} \\ \gamma_{xz} &= \frac{\partial u_x}{\partial z} + \frac{\partial u_z}{\partial x} \\ \gamma_{yz} &= \frac{\partial u_y}{\partial z} + \frac{\partial u_x}{\partial z} \end{aligned} \tag{2.2}$$

We now redefine the strain components in smart way such that system (2.2) can be written in a more compact way. We define

$$e_{ij} = \begin{cases} \varepsilon_i & \text{if } i = j \\ \frac{1}{2}\gamma_{ij} & \text{if } i \neq j \end{cases} \tag{2.3}$$

So that (2.2) can be rewritten as

$$e_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) \qquad \text{for all } 1 \leq i, j \leq 3 \tag{2.4}$$

or even shorter

$$e = \frac{1}{2}\left(\frac{\mathrm{d}\mathbf{u}}{\mathrm{d}\mathbf{x}} + \left(\frac{\mathrm{d}\mathbf{u}}{\mathrm{d}\mathbf{x}}\right)^T\right) \tag{2.5}$$

where $\frac{\mathrm{d}\mathbf{u}}{\mathrm{d}\mathbf{x}}$ is the Jacobian matrix of $\mathbf{u}$.

2.4 Stress

Stress is the physical quantity that represents the force per unit square that is being extered on a particle in a body. The unit of stress is therefore Nm^{-2}. If there is stress, it means that the material is under tension, so that the forces are attempting to stretch (or indent) the material. Like strain, there are nine stress components, which we denote by σ_{ij} for $1 \leq i, j \leq 3$ (or σ_{xy}, etc). The components σ_{xx}, σ_{yy}, and σ_{zz} are the normal stresses, the others being the shearing stresses (often denoted as τ). Similarly for the strain, the shearing stresses are symmetric (which is a result of the conservation of angular momentum). We can define a stress tensor σ as

$$\sigma = \begin{pmatrix} \sigma_{xx} & \tau_{xy} & \tau_{xz} \\ \tau_{yx} & \sigma_{yy} & \tau_{yz} \\ \tau_{zx} & \tau_{zy} & \sigma_{zz} \end{pmatrix} \tag{2.6}$$

Using this tensor we also define the so-called traction vector $\mathbf{T}$ at the boundary of the domain by

$$\mathbf{T} = \sigma \mathbf{n} \tag{2.7}$$

where $\mathbf{n}$ is the normal vector pointing out of the domain. This traction vector will be used for the boundary conditions.

2.5 Relation between stress & strain

The stress is easily computed, but it is usually the strain we are interested in. We will therefore look at the relation between the stress and the strain. We assume throughout the whole article that all materials are isotropic, that is, the material has no preferred direction; regardless of the direction in which the force is applied, a force will always give the same displacements relative to its direction.

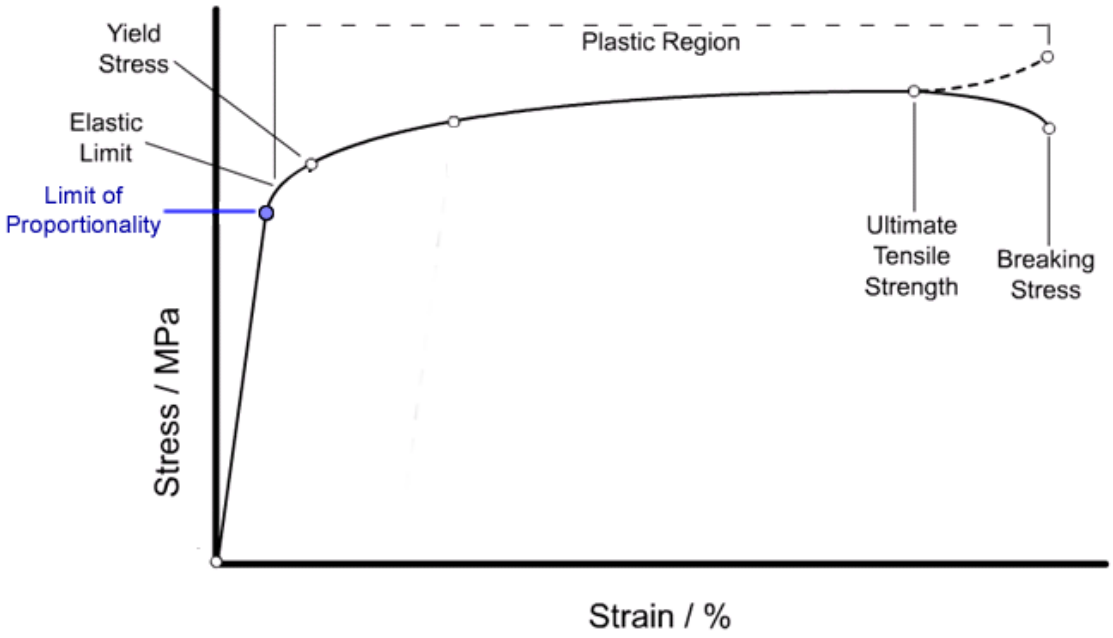

Figure 1: The relation between strain and stress. Source: [3].

For many ductile materials, the relation between stress and strain starts out to be linear until a certain strain ε_0 (also called the limit of proportionality) is reached, see also Figure 1. In linear elasticity theory, we assume that the strains do not surpass this limit of proportionality. According to Hook's law, we have the relation [2]

$$\begin{aligned}
\sigma_{xx} &= \lambda(e_{xx} + e_{yy} + e_{zz}) + 2Ge_{xx} \\
\sigma_{yy} &= \lambda(e_{xx} + e_{yy} + e_{zz}) + 2Ge_{yy} \\
\sigma_{zz} &= \lambda(e_{xx} + e_{yy} + e_{zz}) + 2Ge_{zz} \\
\sigma_{xy} &= Ge_{xy} \\
\sigma_{xz} &= Ge_{xz} \\
\sigma_{yz} &= Ge_{yz}
\end{aligned} \tag{2.8}$$

where λ is Lamé's first parameter, and G the shear modulus. This system of equations can also be written as

$$\sigma_{ij} = \lambda e_{kk}\delta_{ij} + 2Ge_{ij} \quad \text{for } 1 \leq i, j \leq 3 \tag{2.9}$$

where Einstein's convention is used (to sum over k, in this case), and δ being Kronecker's delta function. Equivalently, this is the same as

$$\sigma = \lambda \mathrm{Tr}(e)I + 2Ge \tag{2.10}$$

2.6 Other elastic properties

Other elastic moduli are the bulk modulus, Young's modulus, Poisson's ratio, and the P-wave modulus. For homogeneous isotropic materials, however, these variables are dependent. Each variable can be computed if any two of the moduli is known. Usually, we use the Young's modulus, also known as the modulus of elasticity, which is denoted by E. Furthermore, we use the Poisson's ratio

A material like steel requires more stress to be exerted to deform a certain distance compared with rubber. The modulus of elasticity of steel ($2{\cdot}10^8$Pa) is much larger than for rubber (approximately $5 \cdot 10^4$Pa). It is defined as

$$E = \frac{G(3\lambda + 2G)}{\lambda + G} \tag{2.11}$$

When a material is compressed in one direction, it usually not only deforms in this direction but also expands in the other two directions perpendicular to the direction of compression. This is called the Poisson effect. This effect depends on the so called Poisson ratio of the material. This dimensionless quantity is denoted by ν and is equal to

$$\nu = \frac{\lambda}{2(\lambda + G)} \tag{2.12}$$

2.7 The equations of motion

The equations of motion form another important aspect in (dynamic) elasticity theory. Around each particle, we compute the body force (such as gravity) in each direction. According to Newton's equation of motion, this is equal to the mass (or in this case, the density ρ of the material around the particle) multiplied by the acceleration of the particle in that direction. It can be shown that the corresponding equations of motions are [2]

$$\begin{aligned}
\frac{\partial \sigma_{xx}}{\partial x} + \frac{\partial \sigma_{xy}}{\partial y} + \frac{\partial \sigma_{xz}}{\partial z} + F_x &= \rho \frac{\partial^2 u_x}{\partial t^2} \\
\frac{\partial \sigma_{yx}}{\partial x} + \frac{\partial \sigma_{yy}}{\partial y} + \frac{\partial \sigma_{yz}}{\partial z} + F_y &= \rho \frac{\partial^2 u_y}{\partial t^2} \\
\frac{\partial \sigma_{zx}}{\partial x} + \frac{\partial \sigma_{zy}}{\partial y} + \frac{\partial \sigma_{zz}}{\partial z} + F_z &= \rho \frac{\partial^2 u_z}{\partial t^2}
\end{aligned} \tag{2.13}$$

which corresponds to

$$\frac{\partial \sigma_{ij}}{\partial x_j} + F_i = \rho \frac{\partial^2 u_i}{\partial t^2} \tag{2.14}$$

or by using tensor notation

$$\nabla \cdot \sigma + \mathbf{F} = \rho \frac{\partial^2 \mathbf{u}}{\partial t^2} \tag{2.15}$$

where $(\nabla \cdot \sigma)_i = \frac{\partial \sigma_{ij}}{\partial x_j}$ (as a regular matrix product, but with the derivatives in front).

By substituting (2.2) in (2.8), and then substituting the resulting expressions for σ in (2.13), we arrive at the alternative equations of motion

$$\begin{aligned}
G\Delta u + (\lambda + G)\frac{\partial}{\partial x}\left(\frac{\partial u_x}{\partial x} + \frac{\partial u_y}{\partial y} + \frac{\partial u_z}{\partial z}\right) + F_x &= \rho \frac{\partial^2 u_x}{\partial t^2} \\
G\Delta v + (\lambda + G)\frac{\partial}{\partial y}\left(\frac{\partial u_x}{\partial x} + \frac{\partial u_y}{\partial y} + \frac{\partial u_z}{\partial z}\right) + F_y &= \rho \frac{\partial^2 u_y}{\partial t^2} \\
G\Delta w + (\lambda + G)\frac{\partial}{\partial z}\left(\frac{\partial u_x}{\partial x} + \frac{\partial u_y}{\partial y} + \frac{\partial u_z}{\partial z}\right) + F_z &= \rho \frac{\partial^2 u_z}{\partial t^2}
\end{aligned} \tag{2.16}$$

or alternatively

$$G\frac{\partial^2 u_i}{\partial x_j^2} + (\lambda + G)\frac{\partial^2 u_j}{\partial x_i x_j} + F_i = \rho \frac{\partial^2 u_i}{\partial t^2} \qquad \text{for } 1 \leq i \leq 3 \tag{2.17}$$

or in vector notation as

$$G\Delta \mathbf{u} + (\lambda + G)\nabla(\nabla \cdot \mathbf{u}) + \mathbf{F} = \rho \frac{\partial^2 \mathbf{u}}{\partial t^2} \tag{2.18}$$

2.8 The boundary value problem

Equation (2.18) is not complete without specifying the boundary conditions as well as the initial values. There are two main boundary conditions, conditions for the traction and conditions for the displacement. Both can be used at the same time for a different boundary.

Traction boundary conditions should be applied on a boundary if a force is acting on the surface. For example, if a constant pressure $\mathbf{p}_0$ is being exerted on a boundary Γ_1, then the corresponding boundary condition should be $-\mathbf{p}_0 = \mathbf{T} = \sigma \cdot \mathbf{n}$ on Γ_1.

It is also possible to specify the displacement at a boundary. If a boundary Γ_2, for example, is attached to a solid wall, the displacement should be zero. This corresponds to the boundary condition $\mathbf{u} = \mathbf{0}$ on Γ_2.

Finally, the initial values should be specified. Since equation (2.18) is of second order, both initial displacements $\mathbf{u}$ as well as the velocity for the displacements $\dot{\mathbf{u}}$ should be specified.

2.9 The different contact models

For systems with multiple bodies, the situation is more complex. The bodies can have different elastic properties and the pressure distribution at the contact locations is not specifically known. Furthermore, linear elasticity theory does not necessarily describe reality perfectly.

The first real contact model which describes contact mechanics accurately, is the Hertz model, developed in 1880. This theory describes the contact between two elastic spheres (or a sphere and a plane), or more general, between two bodies with a distance $h(\mathbf{x})$, with h being a quadratic function. In this model creep is neglected, i.e. there is no friction between the two objects (which is not completely realistic if the two objects have different elastic properties). Using the Hertz equations the displacements of the materials can be computed. The Hertz model is still being used as of today, although it is still not perfectly realistic.

It took a long time (until the 1960s) till research showed that the Hertz model wasn't completely accurate. In 1970 an improvement of the Hertz model was made. This new model is called the JKR model (named by its inventors Johnson, Kendall, and Roberts). This models also takes adhesion into account; materials that are close to each other experience van der Waals forces to each other. This attraction property is being used in the JKR model.

The MD (Maugis-Dugdale) model is a further improvement of the JKR model. This model includes the effect of plastic deformation.

3 Numerical methods for dynamical problems

3.1 Newton's equation of motion for contact problems

For contact mechanics, Newton's equations of motion play an important role. As a simple one dimensional example, consider a point with mass m on an elastic surface. Gravity exerts a force of mg downwards. The surface will indent slightly, so that the surface will exert a contact force F_c upwards. This force, however, depends on the total penetration, a larger strain requires a larger force as described in Section 2.6. The resulting force therefore is $F(z) = F_c(z) - F$. By Newton's second law of motion, this resulting force is proportional to the acceleration of the point, i.e.

$$F(z) = m\frac{\mathrm{d}^2 z}{\mathrm{d}t^2}$$

This ordinary second order differential equation can be non-linear. Furthermore, this model assumes no friction at the contact and also no air resistance. In general, linear structural dynamic problems in multiple dimensions have the form

$$M\ddot{\mathbf{x}} + C\dot{\mathbf{x}} + K\mathbf{x} = \mathbf{F}(t) \tag{3.1}$$

where M represents the mass matrix, C the stiffness, and K the damping matrix. F is the vector of external forces which depends on the time variable.

In many contact problems, however, the equation of motion is non-linear. The contact force itself is generally non-linear, as we will describe in Section 4. The most general form of the Newton's equation of motion is

$$M(\mathbf{x})\ddot{\mathbf{x}} + P(\mathbf{x}, \dot{\mathbf{x}}) = \mathbf{F}(t) \tag{3.2}$$

with M and P being any function.

3.2 Euler Forward / Backward

To solve differential equation (3.2), integration schemes are required. For simplicity and completeness, we will start with the well known Euler Forward and Backward integration scheme. This scheme allows to integrate first order ordinary differential equations with respect to time. To apply this, we first transform equation (3.2) to a (twice as large) system of first order differential equations. We introduce $\mathbf{y} = \dot{\mathbf{x}}$ so that

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{y} \\ \dot{\mathbf{y}} &= M^{-1}(\mathbf{x})(\mathbf{F}(t) - P(\mathbf{x}, \mathbf{y})) \end{aligned} \tag{3.3}$$

or shortened as $\dot{\mathbf{z}} = \mathbf{g}(t, \mathbf{z})$ for $\mathbf{z} = (\mathbf{x}, \mathbf{y})^T$.

The Euler Forward scheme is the simplest explicit time integration scheme. It is of order $\mathcal{O}(\Delta t)$ and is described by

$$\mathbf{z}_{k+1} = \mathbf{z}_k + \Delta t \mathbf{g}(t_k, \mathbf{z}_k)$$

The Euler Backward scheme is the simplest implicit time integration scheme, also of order $\mathcal{O}(\Delta t)$, and is described by

$$\mathbf{z}_{k+1} = \mathbf{z}_k + \Delta t \mathbf{g}(t_{k+1}, \mathbf{z}_{k+1})$$

Often, because $\mathbf{g}$ can be a complex function, it is not possible to write $\mathbf{z}_{k+1}$ as a simple function if t_{k+1} and $\mathbf{z}_k$. Instead, one needs to apply an iterative scheme to approximate this solution.

Picard iteration is the easiest one to apply. One can set $\mathbf{z}_{k+1}^0 = \mathbf{z}_k$ as a good initial guess and repeatedly compute

$$\mathbf{z}_{k+1}^{j+1} = \mathbf{z}_k + \Delta t \mathbf{g}(t_{k+1}, \mathbf{z}_{k+1}^j)$$

until $\|\mathbf{z}_{k+1}^{j+1} - \mathbf{z}_{k+1}^j\| < \varepsilon$ for a certain error margin. Alternatively, one can apply a faster converging iterative method like Newton-Raphson (or the Secant method if the exact derivatives of $\mathbf{g}$ are not known).

3.3 Runge Kutta / Radau methods

The Runge-Kutta methods are generalizations of the Euler Forward/Backward methods. These methods are used to integrate the same first order ordinary differential equation with respect to time, but with a higher order accuracy.

All Runge-Kutta methods have the form

$$\mathbf{z}_{k+1} = \mathbf{z}_k + \Delta t \sum_{i=1}^{n} b_i \mathbf{k}_i \tag{3.4}$$

where

$$\mathbf{k}_i = \mathbf{g}(t_k + c_i \Delta t, \mathbf{z}_k + \Delta t \sum_{j=1}^{n} a_{ij} \mathbf{k}_j) \tag{3.5}$$

If A, the matrix with coefficients A_{ij}, is a strictly lower triangular matrix, then computing $\mathbf{k}_i$ does not require the use of $\mathbf{k}_j$ for $j \geq i$. Hence the method is explicit. If A is not strictly lower triangular, then the method is implicit.

The components of the matrix A and vectors $\mathbf{c}$ and $\mathbf{b}$ are often compactly written by using a Butcher's tableau. For the widely used RK4 method, for example, we have the tableau

$$\begin{array}{c|cccc} c_1 & a_{11} & a_{12} & \dots & a_{1n} \\ c_2 & a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \vdots & & \vdots \\ c_b & a_{n1} & a_{n2} & \dots & a_{nn} \\ \hline & b_1 & b_2 & \dots & b_n \end{array} \quad = \quad \begin{array}{c|cccc} 0 & 0 & 0 & 0 & 0 \\ 1/2 & 1/2 & 0 & 0 & 0 \\ 1/2 & 0 & 1/2 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ \hline & 1/6 & 1/3 & 1/3 & 1/6 \end{array}$$

Using (3.4) and (3.5), this corresponds to the fully explicit scheme of order $\mathcal{O}(\Delta t^4)$:

$$\begin{aligned} \mathbf{k}_1 &= \Delta t \mathbf{g}(t_k, \mathbf{z}_k) \\ \mathbf{k}_2 &= \Delta t \mathbf{g}(t_k + \frac{1}{2}\Delta t, \mathbf{z}_k + \frac{1}{2}\mathbf{k}_1) \\ \mathbf{k}_3 &= \Delta t \mathbf{g}(t_k + \frac{1}{2}\Delta t, \mathbf{z}_k + \frac{1}{2}\mathbf{k}_2) \\ \mathbf{k}_4 &= \Delta t \mathbf{g}(t_k + \Delta t, \mathbf{z}_k + \mathbf{k}_3) \\ \mathbf{z}_{k+1} &= \mathbf{z}_k + \frac{1}{6}\mathbf{k}_1 + \frac{1}{3}\mathbf{k}_2 + \frac{1}{3}\mathbf{k}_3 + \frac{1}{6}\mathbf{k}_4 \end{aligned}$$

The Radau methods are fully implicit Runge-Kutta methods of order $\mathcal{O}(\Delta t^{2n-1})$ where n is the number of stages. As described in [5], the default Radau5 method is given by the Butcher's tableau

$$\begin{array}{c|ccc} 0 & \frac{1}{9} & \frac{-1-\sqrt{6}}{18} & \frac{-1+\sqrt{6}}{18} \\ \frac{3}{5}-\frac{\sqrt{6}}{10} & \frac{1}{9} & \frac{11}{45}+\frac{7\sqrt{6}}{360} & \frac{11}{45}-\frac{43\sqrt{6}}{360} \\ \frac{3}{5}+\frac{\sqrt{6}}{10} & \frac{1}{9} & \frac{11}{45}+\frac{43\sqrt{6}}{360} & \frac{11}{45}-\frac{7\sqrt{6}}{360} \\ \hline & \frac{1}{9} & \frac{4}{9}+\frac{\sqrt{6}}{36} & \frac{4}{9}-\frac{\sqrt{6}}{36} \end{array}$$

Since the Radau methods are implicit, an iterative scheme should be used to determine the solutions $\mathbf{z}^{k+1}$.

3.4 The Verlet method

The Verlet integration scheme [6] can be used to integrate the second order differential equation of the form $\ddot{\mathbf{x}} = \mathbf{g}(\mathbf{x})$. It is an explicit method of order $\mathcal{O}(\Delta t^2)$ and is given by

$$\begin{aligned} \mathbf{x}_1 &= \mathbf{x}_0 + \mathbf{v}_0 \Delta t + \frac{(\Delta t)^2}{2}\mathbf{g}(\mathbf{x}_0), \text{ and} \\ \mathbf{x}_{k+1} &= 2\mathbf{x}_k - \mathbf{x}_{k-1} + (\Delta t)^2 \mathbf{g}(\mathbf{x}_k) \qquad \text{for } k \geq 1 \end{aligned} \tag{3.6}$$

where $\mathbf{x}_0$ and $\mathbf{v}_0$ are the position and speed at time $t = t_0$. An interesting property of the Verlet method is that out of all the explicit method we have discussed, it is the only energy conserving method. We will show this in Section 4. However, it is not always applicable for the most general case (3.2), for example when there is air resistance, so that $\mathbf{g}$ is also a function of $\dot{\mathbf{x}}$.

3.5 Newmark's method

Newmark's method (also known as the Newmark-beta method) is a popular integration scheme for problems in structural dynamics. The algorithm is described by

$$\begin{aligned} \mathbf{x}_{k+1} &= \mathbf{x}_k + \Delta t \mathbf{v}_k + \frac{(\Delta t)^2}{2}((1-2\beta)\mathbf{a}_k + 2\beta \mathbf{a}_{k+1}) \\ \mathbf{v}_{k+1} &= \mathbf{v}_k + \Delta t((1-\gamma)\mathbf{a}_k + \gamma \mathbf{a}_{k+1}) \end{aligned} \tag{3.7}$$

Here $\mathbf{v}_{k+1}$ and $\mathbf{a}_{k+1}$ are approximations of the velocity $\dot{\mathbf{x}}$ and acceleration $\ddot{\mathbf{x}}$ at time t_{k+1}, respectively. The acceleration $\mathbf{a}_{k+1}$ is derived from the equations of motion (3.2) by

$$M(\mathbf{x}_{k+1})\mathbf{a}_{k+1} + P(\mathbf{x}_{k+1}, \mathbf{v}_{k+1}) = \mathbf{F}(t_{k+1}) \tag{3.8}$$

Both parameters γ and β are required to be in $[0, 1]$. The method is implicit, unless both γ and β are zero. If $\gamma = 1/2$, then the method is of second order accuracy and there is no numerical

damping. If, however, $\gamma \neq 1/2$, then the method is only of order $\mathcal{O}(\Delta t)$. The parameter β defines how the acceleration is interpolated. Usually one takes $\beta = 1/4$, so that the acceleration is averaged between timestep t_k and t_{k+1}. Another possibility is setting $\beta = 1/6$, which assumes that the acceleration is linear in $[t_k, t_{k+1}]$.

3.6 The HHT method

The disadvantage of the Newmark-beta method is that numerical damping can only be introduced by lowering the order of accuracy. With this in mind, the HHT method (named after its inventors Hilber, Hughes, and Taylor), also known as the α-method, was constructed. The HHT method [7] is an extension of the Newmark method. An extra parameter α is introduced. The equations (3.7) remain the same, however, instead of the discrete Newton's equation of motion (3.8) the HHT method is slightly different:

$$M(\mathbf{x}_{k+1})\mathbf{a}_{k+1} + (1-\alpha)P(\mathbf{x}_{k+1}, \mathbf{v}_{k+1}) + \alpha P(\mathbf{x}_k, \mathbf{v}_k) = F((1-\alpha)t_{k+1} + \alpha t_k) \tag{3.9}$$

If $\alpha = 0$, then the term $\alpha P(\mathbf{x}_k, \mathbf{v}_k)$ drops out of (3.10) and the right-hand side will be equal to $F(t_{k+1})$, so that we are left with Newmark's method. The parameters can be modified so that numerical damping can occur. If γ and β are chosen such that $\gamma = (1+2\alpha)/2$ and $\beta = (1+\alpha)^2/4$, then the method is second order accurate.

3.7 The generalized-α method

The generalized-α method [8] is again a generalisation of the HHT method. Once again the equations (3.7) remain the same, but now instead of the parameter α we have two parameters α_M and α_F. The corresponding equation of motion is:

$$\begin{aligned}&(1-\alpha_M)M(\mathbf{x}_{k+1})a_{k+1} + \alpha_M M(\mathbf{x}_k)a_k + \\ &(1-\alpha_F)P(\mathbf{x}_{k+1}, \mathbf{v}_{k+1}) + \alpha_F P(\mathbf{x}_k, \mathbf{v}_k) = F((1-\alpha_F)t_{k+1} + \alpha_F t_k)\end{aligned} \tag{3.10}$$

If $\alpha_M = 0$, then this is the HHT method with parameter $\alpha = \alpha_F$. This method gives an extra degree of freedom. Parameters γ and β are usually chosen as $\gamma = 1/2 - \alpha_M + \alpha_F$ and $\beta = (1 - \alpha_M + \alpha_F)^2/4$.

4 A simple problem

In this section, we discuss a simple contact problem involving a sphere falling on an elastic surface using Hertz theory. The ball is either solid or elastic, but in the latter case we assume that the modulus of elasticity of the ball is the same as for the surface. This is required, since otherwise friction would occur in the problem. In this case the theory would not be realistic any more and we wouldn't be able to compute the elastic deformation for each body.

Not only is the problem useful because of its simplicity, the problems and solutions will also help us to construct solutions of more complex problems. Furthermore, the problem is very easy to understand so that the numerical schemes can easily be applied and discussed.

4.1 Deformation of an elastic half space under stress

First, we look at the situation where there is only one body. We assume that this body is an elastic half space (i.e its area is $\{(x, y, z) : z \leq 0\}$ in the static situation without external forces). Now imagine a force F_z pushing downwards at the origin. Then, according to [1], the displacement caused by this force is:

$$u_x = \frac{1+\nu}{2\pi E}\left[\frac{xz}{r^3} - \frac{(1-2\nu)x}{r(r+z)}\right]F_z \tag{4.1}$$

$$u_y = \frac{1+\nu}{2\pi E}\left[\frac{yz}{r^3} - \frac{(1-2\nu)y}{r(r+z)}\right]F_z \tag{4.2}$$

$$u_z = \frac{1+\nu}{2\pi E}\left[\frac{2(1-\nu)}{r} - \frac{z^2}{r^3}\right]F_z \tag{4.3}$$

where $r^2 = x^2 + y^2 + z^2$.

So for the surface elements, i.e. points $(x, y, z) \in \mathbb{R}^3$ such that $z = 0$, we in particular have

$$u_x = -\frac{(1+\nu)(1-2\nu)x}{2\pi E r^2}F_z \tag{4.4}$$

$$u_y = -\frac{(1+\nu)(1-2\nu)y}{2\pi E r^2}F_z \tag{4.5}$$

$$u_z = \frac{(1-\nu^2)}{\pi E r}F_z \tag{4.6}$$

Note that since $z = 0$ we also have $r^2 = x^2 + y^2$. Equation (4.4) and (4.5) are the similar, as we would expect from the symmetry of the problem. From these equations, it appears that the larger the distance from the origin, the smaller the displacement. We assumed that no friction occurs in the problem, so that only the z-component of the displacement (i.e. equation (4.6) is of importance.

If there are multiple forces of varying magnitude and position, then the resulting displacement is the sum of the individual solutions. Usually, a force is not exerted on a single point but on a certain area A. In this case we are interested in the pressure p, i.e. the force per square unit, that is being exerted on the surface. We assume that the pressure is continuous. Then, using equation (4.6), the z-displacement at a point (x, y) on the surface is given by

$$u_z = \iint_A \frac{1-\nu^2}{\pi E} \cdot \frac{p}{r} \mathrm{d}u\mathrm{d}v = \frac{1}{\pi E^*} \iint_A \frac{p(u,v)}{\sqrt{(u-x)^2+(v-y)^2}} \mathrm{d}u\mathrm{d}v \tag{4.7}$$

where

$$E^* = \frac{E}{1-\nu^2} \tag{4.8}$$

According to the Hertz theory, the pressure has a distribution of the form

$$p(r) = p_0 \left(1 - \frac{r^2}{a^2}\right)^{1/2} \tag{4.9}$$

where a is the radius of the contact area. Note that (4.9) is only defined for $r \leq a$. The total force is therefore

$$F = \int_0^a p(r) 2\pi r \mathrm{d}r = 2\pi p_0 \left[\frac{-1}{3} a^2 \left(1 - \frac{r^2}{a^2}\right)^{3/2}\right]_0^a = \frac{2}{3}\pi p_0 a^2 \tag{4.10}$$

As derived in [4], if we substitute (4.9) into (4.7), we arrive at the displacement

$$u_z = \frac{\pi p_0}{4E^* a}(2a^2 - r^2) \tag{4.11}$$

4.2 Contact between a rigid sphere and an elastic surface

Imagine the contact between a rigid sphere of radius R and an elastic half space. The height of the sphere at radius r is

$$z = R - \sqrt{R^2 - r^2} \approx \frac{r^2}{2R} \tag{4.12}$$

Let d be the total penetration of the sphere on the surface (i.e. the displacement at the point $(x, y) = (0, 0)$). Then the vertical displacement at radius r is approximately

$$u_z = d - (R - \sqrt{R^2 - r^2}) \approx d - \frac{r^2}{2R} \tag{4.13}$$

Equation (4.13) needs to be of the form (4.11). For this to be true, we must have

$$d = \frac{\pi p_0 a}{2E^*} \tag{4.14}$$

and furthermore

$$\frac{1}{2R} = \frac{\pi p_0}{4E^* a} \implies a = \frac{\pi p_0 R}{2E^*} \tag{4.15}$$

A direct consequence is that the contact radius satisfies

$$a^2 = Rd \tag{4.16}$$

and we also can retrieve an explicit formula for p_0:

$$p_0 = \frac{2E^*a}{\pi R} = \frac{2E^*}{\pi R}\sqrt{Rd} = \frac{2E^*}{\pi}\sqrt{\frac{d}{R}} \tag{4.17}$$

And finally by substituting equations (4.16) and (4.17) into (4.10) we get

$$F = \frac{2}{3}\pi p_0 a^2 = \frac{2}{3}\pi\frac{2E^*}{\pi}\sqrt{\frac{d}{R}}Rd = \frac{4}{3}E^*R^{1/2}d^{3/2} \tag{4.18}$$

4.3 Contact between two curved surfaces

Hertz theory can also be applied in the more general problem with two bodies with curved surfaces.

Suppose the curvature of both surfaces is R_1 and R_2, respectively. It appears that [4] equations (4.16) - (4.18) remain true, as long as the radius R is chosen such that

$$\frac{1}{R} = \frac{1}{R_1} + \frac{1}{R_2} \tag{4.19}$$

4.4 Contact between elastic bodies

A simple adjustment can be made to apply the previous theory to the case with two curves surfaces where both bodies are elastic. Suppose the bodies have an elasticity modulus E_1 and E_2, and Poisson's modulus ν_1 and ν_2, respectively. Then if we define

$$\frac{1}{E^*} = \frac{1-\nu_1^2}{E_1} + \frac{1-\nu_2^2}{E_2} \tag{4.20}$$

instead of equation (4.8), then the same formulas remain correct. Note that the penetration d here is the penetration of the two bodies if the deformation of both bodies is neglected.

4.5 Derivation of the differential equation

Equation (4.18) gives a relation between the penetration of the rigid sphere in the elastic half space and the normal contact force. With this in mind, we can easily retrieve the differential equation describing this test problem.

Consider a solid ball of radius R which is dropped from height z_0 above an elastic surface having Young's modulus E and Poisson's ratio ν. At this time, the velocity $v_0 = 0$. If the density of the ball is ρ, then the mass of the ball is simply $m = \frac{4}{3}\pi R^3\rho$.

At all times, there is gravity pointing downwards with force given by $F_g = -mg$. There can also be a contact force pointing upwards. If the height of the ball with respect to the $z = 0$ plane at a certain point in time is z (which is negative if and only if there is penetration), then the penetration is $d_n = \max(0, -z)$, i.e. zero when the surface is not reached, and otherwise $d = -z$. The contact force is therefore by equation (4.18)

$$F_c = \frac{4}{3}E^*R^{1/2}\max(0,-z)^{3/2} \tag{4.21}$$

So that the resulting force is

$$F(z) = F_g + F_c = \frac{4}{3}E^* R^{1/2} \max(0, -z)^{3/2} - mg \tag{4.22}$$

By Newton's second law, $F(z) = m\ddot{z}$. Dividing by m gives

$$\ddot{z} = \frac{4}{3m}E^* R^{1/2} \max(0, -z)^{3/2} - g \tag{4.23}$$

which is a non-linear ordinary differential equation of form (3.2), where $M \equiv 1$, $F \equiv 0$ and $P(z) = g - \frac{4}{3m}E^* R^{1/2} \max(0, -z)^{3/2}$.

4.6 Properties of the solution

It does not seem to be possible to solve differential equation (4.23) analytically, but it is easy to derive some properties of the solution. Let t_0 be the first moment the ball touches the surface. Before this point, equation (4.23) is simply

$$\ddot{z} = -g \tag{4.24}$$

The exact solution of this is of course $z(t) = -\frac{1}{2}gt^2 + c_1 t + c_0$. Furthermore, since $z'(0) = 0$ and $z(0) = z_0$, we find $c_1 = 0$ and $c_0 = z_0$, so that

$$z(t) = -\frac{1}{2}gt^2 + z_0 \tag{4.25}$$

for $0 \leq t \leq t_0$. It follows that $t_0 = \sqrt{2z_0}$. However, after touching the surface, the differential equation becomes non-linear and very hard to solve analytically. By substituting $w := -z$ and defining $a := -\frac{4E^* R^{1/2}}{3m}$ we can retrieve the following by using separation of variables:

$$\begin{aligned}
&\frac{\mathrm{d}^2 w}{\mathrm{d}t^2} = aw^{3/2} + g \\
\implies &w^{-3/2}\frac{\mathrm{d}^2 w}{\mathrm{d}t^2} = a + gw^{-3/2} \\
\implies &w^{-3/2} d^2 w = (a + gw^{-3/2}) dt^2 \\
\implies &\frac{w^{-3/2}}{a + gw^{-3/2}} d^2 w = dt^2
\end{aligned} \tag{4.26}$$

We can now try to integrate equation (4.26) twice. Maple can do this analytically. The left side turns out to be a very large function h made of logarithms and an inverse tangent function. The right side is simply $\frac{1}{2}t^2 + c_1 t + c_0$. Since the left-hand side is so complex, it is not possible to invert this function analytically. Hence w cannot be explicitly defined as function of t.

We do know, however, that at some point $t_1 > t_0$, the ball reaches the lowest point (i.e. its kinetic energy is zero). Since there is no damping in the system and therefore no energy is lost, that the solution is symmetric with respect to $t = t_1$. So at $t = t_1 + (t_1 - t_0)$ the ball will be at height 0 again, and at time $t = 2t_1$ the ball reaches its original height z_0. This cycle repeats indefinitely.

4.7 Solving the problem numerically

Next, we will discuss the numerical methods as described in Section 3 applied for this test problem. All of the methods have been applied for this problem, but we only discuss the findings since many of the results are similar.

The Radau5 and the Verlet method both give the result as we would expect, as can be seen in Figure 2. Furthermore, if for Newmark, HHT, or the generalized-α method the parameters are chosen such that no numerical damping is created on purpose (as described in the corresponding chapters), then the results are similar.

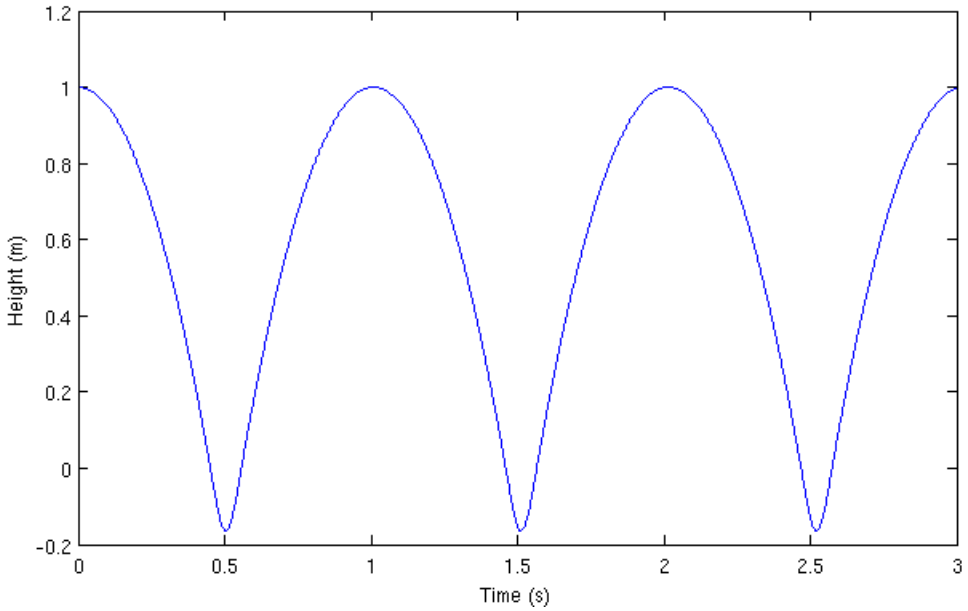

Figure 2: The height of the ball as a function of the time, using a stable method such as Radau5.

It is also interesting to look at the total amount of energy in the system. At $t = 0$, the kinetic and elastic potential energy are both zero, but the potential gravitational energy is maximal. For $t \in [0, t_0]$, the potential gravitational energy is slowly transformed to kinetic energy. Between $t = t_0$ and $t = t_1$, both kinetic and gravitational potential energy are transformed to potential elastic energy. At $t = t_1$, the kinetic energy is 0, the potential gravitational energy is minimal (negative since there is penetration), and the elastic energy potential is maximal.

However, since there is no damping in the system, the total amount of energy (i.e. the sum of the kinetic, gravitational, and elastic energy) is constant. This can be seen in Figure 3.

The fact that the total amount of energy is constant is the reason why explicit methods like Euler Forward and Runge-Kutta 4 are never stable. The effect of Euler Forward can be seen in Figure 4. Note that the height z of the ball is proportional to the total amount of gravitational energy. After each bounce, the total amount of energy is increased. However, if both the time step Δt and the total amount of time t_n are small enough, then this increase of energy will not be visible.

For an implicit Runge-Kutta method like the Euler Backward, the exact opposite happens if Δt is too small. In this case there will be numerical damping, so that the total amount of energy decreases over time.

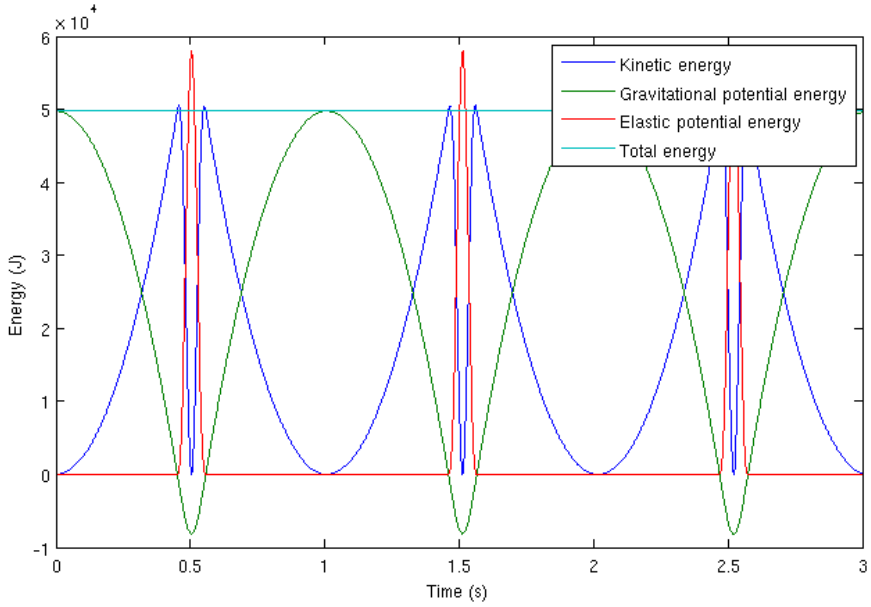

Figure 3: The energy of the ball as a function of the time, using a stable method such as Radau5.

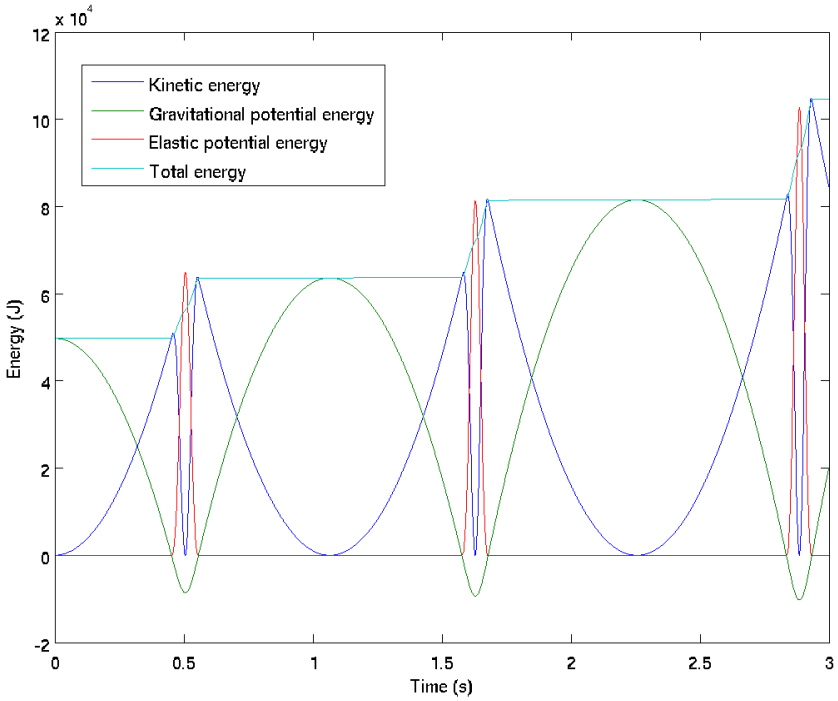

Figure 4: The energy of the ball as a function of the time, using the instable Euler Forward method.

4.8 Using CONTACT to solve the problem

It is interesting if this problem can also be solved by using CONTACT. Instead of using equation (4.18) to determine the contact force, we could also compute the normal force by using CONTACT. Using this we can compare the solution with the exact solution for the Hertz problem, and also see if the properties of each numerical method still hold.

For CONTACT one of the two options should be specified, either contact force or the penetration. For our problem we are interested in the second option. The surface of the ball is discretised on a grid $[X, Y]$. The height Z_{ij} of the sphere at surface element (X_{ij}, Y_{ij}) can easily be computed

$$Z_{ij} = R - \sqrt{R^2 - (X_{ij}^2 + Y_{ij}^2)} + z \tag{4.27}$$

As before, z denotes the height of the ball, measured from its lowest point. Note that equation (4.27) is only valid if Z_{ij} is real. If it is not real, then the line (X_{ij}, Y_{ij}, z) does not intersect with the surface of the ball.

If a value Z_{ij} is negative, then this can be seen as the penetration of the ball in the surface. The penetration Z can be supplied to CONTACT, together with other required variables such as the elasticity coefficients of the materials and the size of the domain. Using this, CONTACT is able to compute the contact force as well as the deformation of the elastic surface outside of the contact area.

We denote the contact force as function of the height z by $F_c(z)$. Note that for $z \geq 0$ the surface is still in its undeformed state, hence $F_c(z) = 0$. Similarly to the derivation of the differential equation (4.23), we arrive the differential equation

$$\ddot{z} = \frac{F_c(z)}{m} - g \tag{4.28}$$

The same numerical methods can be applied as before. Note that the computation of $F_c(z)$ is relatively expensive (unless of course $z \geq 0$) so implicit methods will generally take much longer. An advantage, however, is that by using CONTACT arbitrary shapes can be used instead of just a ball . Furthermore, the displacement of the surface outside of the contact area can be computed with CONTACT.

We programmed the explicit Verlet method and the implicit Radau5 method in Matlab. The results are clearly not correct yet and will therefore require more intensive research. See Figure 6 for a comparison between the CONTACT method and the hertz solution. The most obvious observation is that Verlet method now is very unstable. If we lower the timestep Δt, then this amplification factor will decrease slowly. Radau5, however, seems more accurate. As for the hertz problem, Radau5 makes sure there is energy conservation. For both solutions, however, the maximum penetration is slightly off.

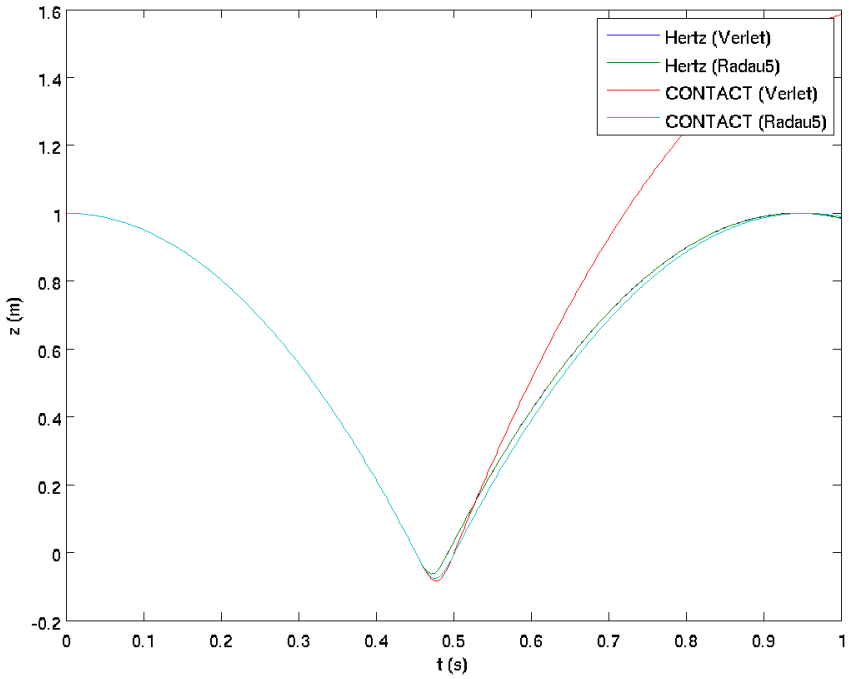

Figure 5: Comparison of the exact hertz solution and the CONTACT solution using the Verlet and the Radau5 numerical integration scheme.

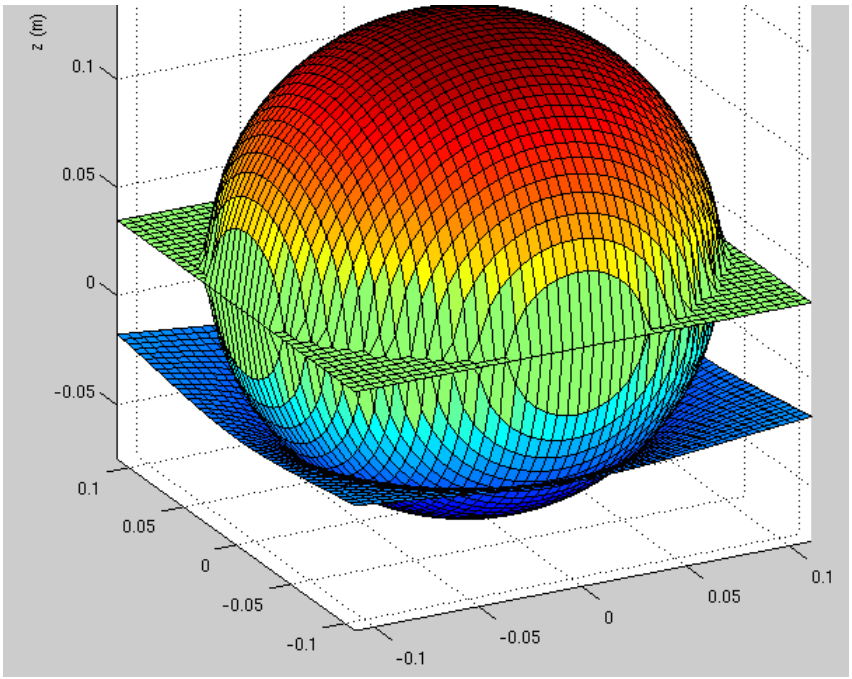

Figure 6: The deformation of the surface at a certain point in time, computed using CONTACT.

5 Global deformations of a beam

In the following more complex dynamic contact problem global deformations are also of importance. Consider an elastic beam with a force acting perpendicularly. This force will not only cause local deformation of the beam around the initial point of contact, the whole beam also gains momentum from the impact, thereby causing global deformation. In particular, we are interested how CONTACT could be applied for these problems.

This example is for example useful when interested in a moving load problem, such as a train riding on a bridge. The bridge can be seen as an elastic beam with fixed ends. Since the train does not stay in one position, the force of the train exerting on the bridge also differs with respect to both place and time.

5.1 The 1D dynamic beam equation

In the following example, we consider a one dimensional beam. The two dimensional case is a simple generalisation of this problem. Euler-Bernoulli beam theory is a simplification of the most general elasticity theory as described in Section 2. Instead of using differential equation (2.18), the Euler-Bernoulli dynamic elastic beam differential equation in one dimension is

$$\frac{\partial^2}{\partial x^2}\left(E(x)I(x)\frac{\partial^2 u}{\partial x^2}\right) = -\rho\frac{\partial^2 u}{\partial t^2} + p(x,t) \tag{5.1}$$

Here E is the elasticity modulus of the material of the beam. The parameter ρ describes the mass per unit length and p corresponds to the force per unit length (their 3D counterparts being density and pressure, respectively). I is the second moment of area of the cross-section of the beam. This property reflects how the points of an object are distributed with respect to a certain axis L and is defined as

$$I(x) = \int_{A(x)} d(z)^2 \mathrm{d}z \tag{5.2}$$

in the one dimensional case, or

$$I(x) = \iint_{A(x)} d(y,z)^2 \mathrm{d}y\mathrm{d}z \tag{5.3}$$

in the two dimensional case. Here $A(x)$ is the (either one or two dimensional) cross section of the beam at x, and d is defined as the distance from this point to the axis L. For the beam, this axis L is simply the axis $y = z = 0$. If the cross-section of the beam has the same distribution and size for each $x \in [0, L]$, then the second moment of area $I(x) = I$ is constant.

The Euler-Bernoulli beam theory neglects shear deformations and the solution is only accurate for long thin beams. More complex (Timoshenko) beam theories have been developed that account for these problems, but we will solely focus on the simpler Euler-Bernoulli beam equation.

When assuming the beam is homogeneous, then both E and I are constant and therefore the following equation can be derived:

$$EI\frac{\partial^4 u}{\partial x^4} = -\rho\frac{\partial^2 u}{\partial t^2} + p(x,t) \tag{5.4}$$

5.2 Boundary and initial conditions

Equation (5.1) is a fourth order differential equation with respect to the variable x, hence we need 4 boundary conditions. Different boundary conditions are possible for the problem, but it is reasonable (if we see the beam as a bridge) to assume that the beam is fixed at $x = 0$ and $x = L$. In this situation, vibration can only occur in $x \in [0, L]$. Outside of this area, the rails are fixed to the ground. Both the displacement u and the angle $\frac{\partial u}{\partial x}$ should be zero at the boundaries:

$$\begin{aligned} u(0,t) = 0 = \frac{\partial u}{\partial x}(0,t) \\ u(L,t) = 0 = \frac{\partial u}{\partial x}(L,t) \end{aligned} \tag{5.5}$$

These conditions are called clamped boundary conditions. Another possibility is to use a simply supported beam. In this situation the bending moment $M = -EI\frac{\partial^2 u}{\partial x^2}$ is assumed to be zero at the boundary, instead of the slope of the beam:

$$\begin{aligned} u(0,t) = 0 = \frac{\partial^2 u}{\partial x^2}(0,t) \\ u(L,t) = 0 = \frac{\partial^2 u}{\partial x^2}(L,t) \end{aligned} \tag{5.6}$$

where EI is assumed to be non-zero.

Furthermore, we have a second derivative for the time variable, so that we need an initial condition for both the displacement u and the velocity $\frac{\partial u}{\partial t}$ of the displacement.

5.3 Modal analysis

5.3.1 Free vibrations

In many situations, we are interested in the frequencies of the solutions of equation (5.4). Consider for example the free vibrations of the beam, these are solutions of the same equation where there is no external force, i.e. $p \equiv 0$. The resulting partial differential equation can be solved using separation of variables (also known as the Fourier method). Let $v(x,t) = \mathrm{Re}[w(x)\mathrm{e}^{i\lambda t}]$. By substituting this into differential equation $EI\frac{\partial^4 u}{\partial x^4} = -\rho\frac{\partial^2 u}{\partial t^2}$, we find that v is a solution if and only if

$$EIw^{(4)}(x)\mathrm{e}^{i\lambda t} = \rho\lambda^2 w(x)\mathrm{e}^{i\lambda t} \tag{5.7}$$

Dividing by $EI\mathrm{e}^{i\lambda t}$ yields the ordinary differential equation

$$w^{(4)}(x) = \frac{\lambda^2\rho}{EI}w(x) \tag{5.8}$$

This differential equation has solutions of the form $w(x) = \cos(\beta x)$, as well as $\sin(\beta x)$, $\cosh(\beta x)$, and $\sinh(\beta x)$. The parameter β much satisfy $\beta^4 = \frac{\lambda^2\rho}{EI}$, and therefore

$$\beta = \left(\frac{\lambda^2 \rho}{EI}\right)^{1/4} \tag{5.9}$$

We write the solution w as

$$w(x,t) = c_1 \cos(\beta x) + c_2 \sin(\beta x) + c_3 \cosh(\beta x) + c_4 \sinh(\beta x) \tag{5.10}$$

Note that if $\beta = 0$, it follows immediately that w is a constant. It must therefore be identically zero because of the boundary conditions. Since we are only looking for non-trivial solutions, we can assume that $\beta \neq 0$. In this case, w satisfies the boundary equation if and only if

$$\begin{aligned}
0 =& v(0) = c_1 + c_3 \\
0 =& \frac{\partial v}{\partial x}(0) = \beta(c_2 + c_4) \\
0 =& v(L) = c_1 \cos(\beta L) + c_2 \sin(\beta L) + c_3 \cosh(\beta L) + c_4 \sinh(\beta L) \\
0 =& \frac{\partial v}{\partial x}(L) = \beta(-c_1 \sin(\beta L) + c_2 \cos(\beta L) + c_3 \sinh(\beta L) + c_4 \cosh(\beta L)
\end{aligned} \tag{5.11}$$

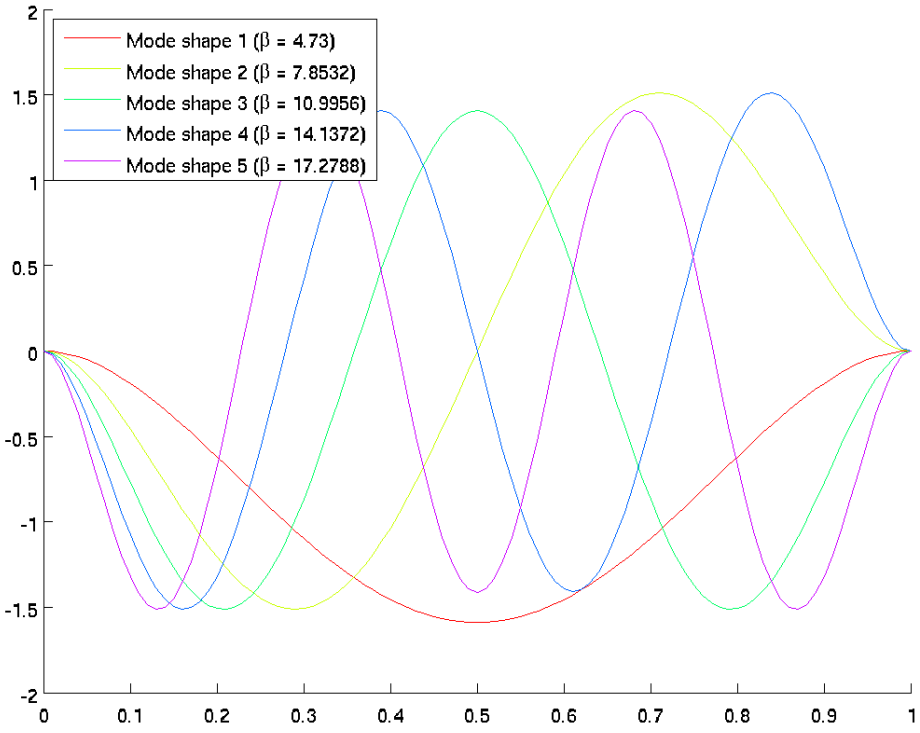

Figure 7: The first five mode shapes of the beam.

By dividing the 3rd and fourth equation by β, this system can be denoted as

$$\begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ \cos(\beta L) & \sin(\beta L) & \cosh(\beta L) & \sinh(\beta L) \\ -\sin(\beta L) & \cos(\beta L) & \sinh(\beta L) & \cosh(\beta L) \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \tag{5.12}$$

or as $A\mathbf{c} = \mathbf{0}$. We are only looking for non-trivial solutions, i.e. solutions $\mathbf{c} \neq \mathbf{0}$. Such a solution can only exist if A is singular (not invertible), i.e. $\det(A) = 0$. A simple computation yields that this is equivalent to

$$\cos(\beta L)\cosh(\beta L) = 1 \tag{5.13}$$

Equation (5.13) has infinitely many solutions and each solution can be approximated by using a numerical method like Newton-Raphson. Since $\cosh(\beta L)$ tends to infinity as $\beta L \to \pm\infty$, $\cos(\beta L)$ must be near zero. Therefore, β_n tends to $\frac{1}{L}(\frac{1}{2} + k)\pi$ for certain $n \in \mathbb{Z}$ as it tends to $\pm\infty$.

The real frequencies λ_n of the beam are easily derived from equation (5.9):

$$\lambda_n = \beta_n^2 \sqrt{\frac{\rho}{EI}} \tag{5.14}$$

The coefficients c_n^i for $i = 1, 2, 3, 4$ can be found by finding a non-zero vector $\mathbf{c}_n$ in the nullspace of A, i.e. $A\mathbf{c}_n = \mathbf{0}$, using the corresponding value of β_n. Note that $\text{Rank}(A) \geq 3$ and equality holds if and only if (5.13) is true, so that this vector is unique up to a constant.

For this problem with clamped boundaries, w_n can be computed analytically. After some computation we find the following expression:

$$w_n(x) = \cosh(\beta_n x) - \cos(\beta_n x) + k_n \sin(\beta_n x) - k_n \sinh(\beta_n x) \tag{5.15}$$

and any multiple of this is also a solution. In this formula k_n is defined as

$$k_n = \frac{\sin(\beta_n L)\sinh(\beta_n L)}{\cos(\beta_n L)\sinh(\beta_n L) - \sin(\beta_n L)\cosh(\beta_n L)} \tag{5.16}$$

The functions w_n are called the mode shapes of the beam with corresponding frequencies λ_n. As an example, suppose $E = I = \rho = L = 1$. Using Newton-Raphson to solve (5.13), we find the following first 5 values of β_n and λ_n as in Table 1. The second column contains the approximation of β_n given by $\frac{1}{L}(\frac{1}{2} + n)\pi$.

n	β_n	$\frac{1}{L}(\frac{1}{2} + n)\pi$	λ_n
1	4.73004	4.71238	22.373
2	7.85320	7.85398	61.672
3	10.9956	10.9955	120.90
4	14.1371	14.1371	199.85
5	17.2787	17.2787	298.55

Table 1: The first 5 frequencies of the beam using $E = I = \rho = L = 1$.

The approximation $\frac{1}{L}(\frac{1}{2}+n)\pi$ therefore converges very quickly to the value of β_n. Note that $\beta = 0$ also satisfies (5.13), but this corresponds to a zero modal function. Furthermore, since $\lambda_n = \beta_n^2\sqrt{\frac{\rho}{EI}}$, we can ignore negative values of n too.

Using superposition the solution u can be written as

$$u(x,t) = \sum_{i=1}^{\infty} v_i(x,t) = \sum_{i=1}^{\infty} \mathrm{Re}[d_i w_i(x) \mathrm{e}^{i\lambda t}] \tag{5.17}$$

The coefficients d_i are still unknown should be computed by using the initial conditions. Note that these coefficients can be complex. The exponential could be expanded and each coefficient d_i can be replaced for two real coefficients a_i and b_i. We find

$$u(x,t) = \sum_{i=1}^{\infty} v_i(x,t) = \sum_{i=1}^{\infty} w_i(x)[a_i \cos(\lambda t) + b_i \sin(\lambda t)] \tag{5.18}$$

The coefficients a_i and b_i can now be extracted by applying the initial conditions and by noting that the mode shapes w_i are orthogonal. Suppose for example that at $t = 0$ there is a certain displacement $u(x,0) = u_0(x)$ but the beam is at rest, i.e. $\dot{u}(x,0) \equiv 0$. Then it is easy to see that $b_i = 0$ for all i. The coefficients m_i could next be computed by using the orthogonality of the set $\{w_i : i \geq 1\}$:

$$a_i = \frac{2}{L}\int_0^L u_0(x) w_i(x) \mathrm{d}x \tag{5.19}$$

5.3.2 Forced vibrations

For the moving load problem, the external force $p(x,t)$ is non-zero. This force can trigger vibrations of a different frequency.

Suppose as an example that this force is harmonic, so that it can be written as $p(x,t) = f(x)\cos(\omega t)$ for ω a frequency. f can be any function, in particular we are interested in a single point force, i.e. $f(x) = p_0\delta(x - x_0)$ describes a force p_0 at point $x = x_0$ and 0 everywhere else. An interesting question is how the solution will behave and what frequencies will become dominant.

For general functions of p, the solution of differential equation (5.4) can not be found analytically. In many situations, however, the solution can be approximated by using the modes found in the case without external forces. Suppose we use a finite amount, say N, of nodes to construct our solution. Using superposition u can be written as

$$u(x,t) \approx u_N(x,t) = \sum_{i=1}^{N} d_i(t) v_i(x,t) = \sum_{i=1}^{N} w_i(x)[a_i(t)\cos(\lambda t) + b_i(t)\sin(\lambda t)] \tag{5.20}$$

Since each function $v_i(x,t)$ satisfies the homogeneous boundary conditions, so does u_N. The coefficients a_i and b_i can depend on time. In many situations, we are interested in computing these coefficients to see what frequencies become dominant. Using modes to solve the problem has the advantage that the coefficients are computed to construct u_N and are therefore immediately known.

5.4 The discrete problem

If there is no external force p, then the exact solution can be computed using equation (5.18). A simple approximation can be found by using a limited number of modes, i.e. by settings $a_i = b_i = 0$ for $|i| \geq k$ for some $k \in \mathbb{N}$.

For a more general force p, the problem is harder. An approximation can be found by discretising differential equation (5.4). Both the Finite Element method and the Finite Difference method can be applied for this problem. Since the geometry of a one dimensional beam is very simple, the Finite Difference method is sufficient. Suppose we use a uniform grid of $n+1$ grid points in total, which includes the grid points x_0 and x_n of both boundaries. The position of the grid points are given by $x_i = i\Delta x$ for $0 \leq i \leq n$, where $\Delta x = \frac{L}{n}$.

Since there is a second order time derivative in differential equation (5.4), we should first convert it to a system of two first order differential equations. To do so, we define $v = \frac{\partial u}{\partial t}$ as the speed of the displacement of the beam. By substituting this into (5.4) and rearranging the terms a bit, we arrive at the system of equations

$$\begin{aligned} \frac{\partial u}{\partial t} &= v \\ \frac{\partial v}{\partial t} &= -\frac{EI}{\rho}\frac{\partial^4 u}{\partial x^4} + \frac{p(x,t)}{\rho} \end{aligned} \tag{5.21}$$

For convention, we denote u_j^k as a shorthand notation of $u(x_j, t_k)$. The fourth order derivative $\frac{\partial^4}{\partial x^4}u_j^k = \frac{\partial^4 u}{\partial x^4}(x_j, t_k)$ can be approximated using the following scheme:

$$\frac{\partial^4}{\partial x^4}u_j(t) = \frac{u_{j-2}(t) - 4u_{j-1}(t) + 6u_j(t) - 4u_{j+1}(t) + u_{j+2}(t)}{\Delta x^4} + \mathcal{O}(\Delta x^2) \tag{5.22}$$

We have two kinds of boundary conditions. The Dirichlet boundary conditions $u(0,t) = u(L,t) = 0$ can simply be applied by substituting 0 for each term u_0^k or u_n^k. Alternatively, one could set u_0^k and u_i^k at zero manually at each time step.

For the other boundary conditions, the Neumann conditions $\frac{\partial u}{\partial x}(0,t) = \frac{\partial u}{\partial x}(L,t)$ for all t, we temporarily introduce two virtual points $x_{-1} = -\Delta x$ and $x_{n+1} = (n+1)\Delta x = L + \Delta x$. The derivative $\frac{\partial u}{\partial x}$ at grid point x_k is discretised by $\frac{u_{k+1} - u_{k-1}}{2\Delta x}$. Since this derivative is zero at $x = 0$ (i.e. when $k = 0$), this yields $u_{-1} = u_1$. So if we substitute u_1 for each term u_{-1} and similarly for the other boundary, we have eliminated the virtual points outside our domain.

After discretising, we arrive at the system of equation

$$\begin{aligned} \frac{\mathrm{d}\mathbf{u}}{\mathrm{d}t} &= \mathbf{v} \\ \frac{\mathrm{d}\mathbf{v}}{\mathrm{d}t} &= A\mathbf{u} + \mathbf{f} \end{aligned} \tag{5.23}$$

Next, an appropriate time integration scheme should be used, such as the ones described in Section 3, in order to find an approximation of the solution.

5.4.1 The CFL condition

A very important mathematical property to consider for the discrete problem is the CFL number and the CFL condition. If this condition is not satisfied, then information does not propagate fast enough, creating artificial wiggles in the solution that tend to infinity, thereby making the results useless.

The CFL condition usually occurs in parabolic and hyperbolic differential equations. Although the beam equation is neither parabolic nor hyperbolic, for this differential equation the CFL number also plays a role.

Consider a fully discretised model, where the partial derivative $\frac{\partial^2}{\partial t^2}u_j(t_k)$ is discretised using

$$\frac{\partial^2}{\partial t^2}u_j^k = \frac{u_j^{k+1} - 2u_j^k + u_j^{k-1}}{\Delta t^2} + \mathcal{O}(\Delta t^2) \tag{5.24}$$

If there are no external forces, i.e. $p \equiv 0$, then the discretised differential equation can be written as

$$u_j^{k+1} = \Delta t^2 \frac{EI}{\rho}\left[\frac{-u_{j-2}^k + 4u_{j-1}^k - 6u_j^k + 4u_{j+1}^k - u_{j+2}^k}{\Delta x^4}\right] + 2u_j^k - u_j^{k-1} \tag{5.25}$$

Note that equation (5.25) is equal to the definition of the Verlet integration method. Von Neumann stability analysis can now be applied on this equation.

Since this is a multistep scheme, we should apply its corresponding stability theorem. Let $u_j^k = g^k \mathrm{e}^{ij\xi}$ (i being the imaginary unit here), where $g \neq 0$ is an amplification factor. By substituting this into equation (5.25) and expanding the exponent, we find

$$g^{k+1}\mathrm{e}^{ij\xi} = \frac{EI\Delta t^2}{\rho\Delta x^4} g^k \mathrm{e}^{ij\xi}\left[-\mathrm{e}^{-2i\xi} + 4\mathrm{e}^{-i\xi} - 6 + 4\mathrm{e}^{i\xi} - \mathrm{e}^{2i\xi}\right] + 2g^k\mathrm{e}^{ij\xi} - g^{k-1}\mathrm{e}^{ij\xi} \tag{5.26}$$

Dividing by $g^{k-1}\mathrm{e}^{ij\xi}$ yields

$$g^2 = \frac{EI\Delta t^2}{\rho\Delta x^4} g\left[-\mathrm{e}^{-2i\xi} + 4\mathrm{e}^{-i\xi} - 6 + 4\mathrm{e}^{i\xi} - \mathrm{e}^{2i\xi}\right] + 2g - 1 \tag{5.27}$$

Note that $\mathrm{e}^{i\xi} + \mathrm{e}^{-i\xi} = 2\cos(\xi)$, so we can rewrite this to

$$g^2 + \left(-2 - \frac{EI\Delta t^2}{\rho\Delta x^4}\left[-2\cos(2\xi) + 8\cos(\xi) - 6\right]\right) g + 1 = 0 \tag{5.28}$$

Hence, there are two solutions for g. Let $y = \frac{EI\Delta t^2}{\rho\Delta x^4}[\cos(2\xi) - 4\cos(\xi) + 3]$, then we find

$$g_\pm = \frac{2 - 2y \pm \sqrt{(-2+2y)^2 - 4}}{2} = 1 - y \pm \sqrt{y^2 - 2y} \tag{5.29}$$

Note that $g_\pm$ is allowed to be imaginary, that is, if $y^2 - 2y < 0$, then $g_\pm = 1 - y \pm i\sqrt{2y - y^2}$.

Both amplification factors g_+ and g_- should be less or equal to one in absolute value for all values of ξ. First, we note that $0 \leq \cos(2\xi) - 4\cos(\xi) + 3 \leq 8$, so that $y \geq 0$. If $y > 2$, then $y^2 > 2y$, so that $g_- = 1 - y - \sqrt{y^2 - 2y} \geq 1 - y < -1$, so that the method is unstable. However, if $0 \leq y \leq 2$, then we have

$$|g_\pm| = |1 - y \pm i\sqrt{2y - y^2}| = \sqrt{(1-y)^2 + |2y - y^2|} = \sqrt{1 - 2y + y^2 + 2y - y^2} = 1 \tag{5.30}$$

hence we have stability. Since furthermore $0 \leq \cos(2\xi) - 4\cos(\xi) + 3 \leq 8$ holds, we also find

$$y = \frac{EI\Delta t^2}{\rho \Delta x^4} \leq \frac{2}{\cos(2\xi) - 4\cos(\xi) + 3} = \frac{1}{4} \tag{5.31}$$

which is therefore the CFL condition for the fully discretised model, i.e. the Verlet method. Note that this CFL condition does not only depend on which integration scheme is used, it also depends on how the fourth order derivative $\frac{\partial^4 u}{\partial x^4}$ is discretised. If instead of (5.22) a better approximation is made, the coefficients for the CFL condition can differ.

This is a fairly strong restriction. If the spatial mesh is required to be twice as fine as it was before, then the time step should be set in the order of 4 times smaller, therefore requiring in the order of 4 times more time integration steps.

5.4.2 Implicit methods

As we have seen, explicit time integration methods can result into strong CFL conditions. It might therefore be a good idea to apply an implicit instead. Consider the Backward Euler method. This integration method is unconditionally stable, so that there is no CFL condition. Picard iteration, however, can result in problem for implicit time integration schemes. In general, Picard iteration can stop converging if Δt is too large compared to Δx.

A different iterative solver should be used instead, such as Newton-Raphson. This solver could be combined with a line search algorithm so that it will converge more often.

For the simple implicit Backward Euler method, we can compute the solution analytically. A single Backward Euler step is given by

$$\begin{aligned} \mathbf{u}^{k+1} &= \mathbf{u}^k + \Delta t \mathbf{v}^{k+1} \\ \mathbf{v}^{k+1} &= \mathbf{v}^k + \Delta t[\mathbf{A}^{k+1} + \mathbf{f}^{k+1}] \end{aligned} \tag{5.32}$$

If we define $\mathbf{z}^k = [\mathbf{u}^k; \mathbf{v}^k]$, this system can be rewritten as

$$\mathbf{z}^{k+1} = \mathbf{z}^k + \Delta t B \mathbf{z}^{k+1} + \Delta t \mathbf{g}^{k+1} \tag{5.33}$$

where

$$B = \begin{pmatrix} 0 & I \\ A & 0 \end{pmatrix} \qquad \text{and } \mathbf{g}^{k+1} = \begin{pmatrix} \mathbf{0} \\ \mathbf{f}^{k+1} \end{pmatrix} \tag{5.34}$$

Therefore, $\mathbf{z}^{k+1}$ can be computed analytically:

$$\mathbf{z}^{k+1} = (I - \Delta t B)^{-1}(\mathbf{z}^k + \Delta t \mathbf{g}^{k+1}) \tag{5.35}$$

If the inverse of $(I - \Delta t B)$ is computed, then $\mathbf{u}^{k+1}$ and $\mathbf{v}^{k+1}$ can be computed explicitly at each time step.

5.5 A simple moving load problem

Consider the following (very simplistic) problem with a train (modelled as a single point force) moving over a track. In this problem we do not take local deformations into account, and therefore we can also ignore phenomena such as slip.

Consider a small bridge that spans a total of 20m which is clamped at both sides. There are no supports between the two ends of the bridge. The bridge itself is made out of steel with a thickness of 50cm. An NS DD-AR train with a mass of 76 tons moves over the the bridge with a speed of 30m/s. This example corresponds to the values as shown in Table 2.

Property	Value	Explanation
E	$2 \cdot 10^{11}$	Elastic modulus of steel
I	0.0104m^4	$I(x) = \int_{-0.25}^{0.25} \lvert z \rvert^2 \mathrm{d}z$
L	20m	Bridge of 20 meters
ρ	7900kg/m^3	Density of steel
m	$76 \cdot 10^4\text{kg}$	Mass of train
v	30	Speed of train in m/s

Table 2: The properties of the bridge and train.

The force the train exerts on the bridge mg at the position of the train, which is $x(t) = vt$ and is therefore

$$p(x,t) = mg\delta(x - vt) \qquad 0 \leq x \leq L \tag{5.36}$$

After defining $w = \frac{\partial u}{\partial t}$ and rewriting equation (5.4) as a system of two linear ordinary differential equations, it can then be discretised using a finite difference approach to approximate $\dot{u}$ and $\dot{w}$. Next, a time integration scheme such as the ones we have described in Section 3 can be used to integrate with respect to time. Alternatively, the partial differential equation can also be discretised completely (including the time derivative) as we have done in the previous section.

n	β_n	$\frac{1}{L}(\frac{1}{2} + n)\pi$	λ_n	Frequency $(\lambda_n/(2\pi))$
1	0.23650	0.23561	$0.1089 \cdot 10^{-3}$	$0.1733 \cdot 10^{-4}$Hz
2	0.39266	0.39269	$0.3002 \cdot 10^{-3}$	$0.4778 \cdot 10^{-4}$Hz
3	0.54978	0.54977	$0.5886 \cdot 10^{-3}$	$0.9368 \cdot 10^{-4}$Hz
4	0.70685	0.70685	$0.9730 \cdot 10^{-3}$	$1.5485 \cdot 10^{-4}$Hz
5	0.86393	0.86393	$1.4534 \cdot 10^{-3}$	$2.3132 \cdot 10^{-4}$Hz

Table 3: The first 5 eigenfrequencies of the bridge.

Table 3 shows the corresponding values of λ_n and β_n for the free vibration modes, as well as the corresponding frequencies.

The algorithm to solve this problem (with the properties as shown in Table 2) has been implemented in Matlab. We use a uniform mesh in the spatial direction with mesh size $\Delta x = \frac{L}{100}$.

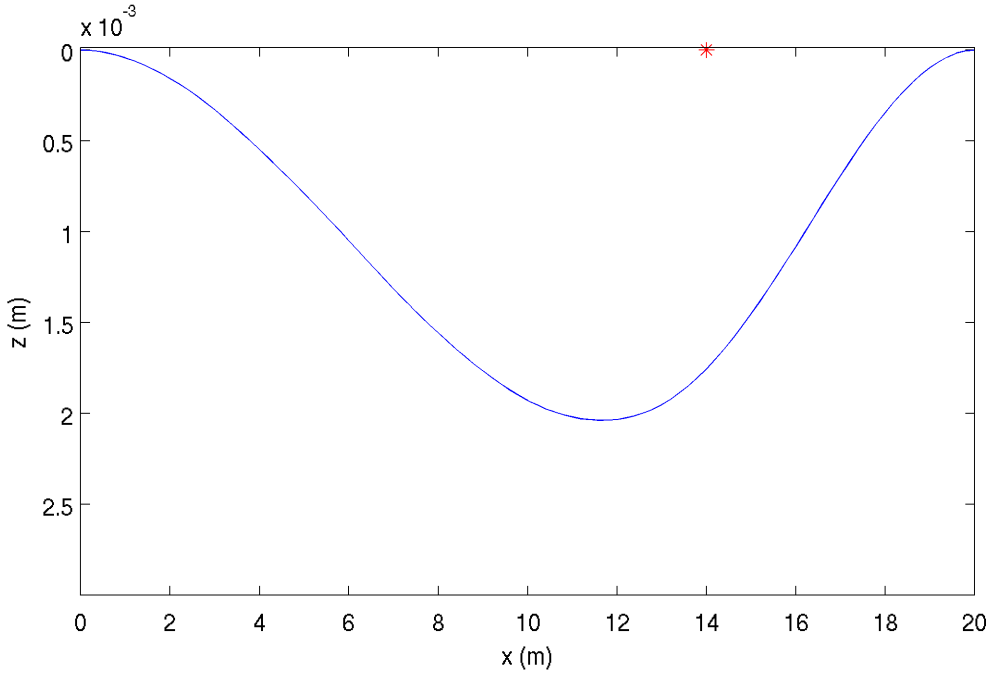

Figure 8: The displacement of the beam at time $t \approx 4.67$. The red star represents the load of the train.

The time integration is done using Verlet integration. According to (5.37), the following CFL condition should hold:

$$\Delta t \leq \frac{(\Delta x)^2}{2}\sqrt{\frac{\rho}{EI}} \approx 3.8977 \cdot 10^{-5} \tag{5.37}$$

In Figure 8 the solution is shown at a certain point of time, using the time step $\Delta t = 3.8976 \cdot 10^{-5}$. The integration is clearly stable. The train moves from the left side to the right side, so as we would expect, a wave appears and follows the train to the right. After applying Verlet integration numerically, it appears that the maximum displacement of the beam occurs at about $t \approx 3.33$. At this time, the train is exactly in the middle of the beam. The total displacement is about 0.3cm.

To show the effect of the CFL condition, we use the same experiment but now with a time step $\Delta t = 3.899 \cdot 10^{-5}$, resulting in a CFL number slightly higher than the upper bound we have found. See the results in Figure 9.

As we can see, artificial wiggles are created which tend to infinity. So our CFL condition is very accurate.

The CFL condition (5.37) was computed for the fully discretised model, i.e. the Verlet method. The Euler Forward method gives a similar result. However, this condition does not seem to be accurate when using the Runge-Kutta 4 method. For the RK4 method, the following CFL condition seems to hold:

$$\Delta t \leq 0.188\frac{\rho}{EI}\Delta x^4 = \mathcal{O}(\Delta x^4) \tag{5.38}$$

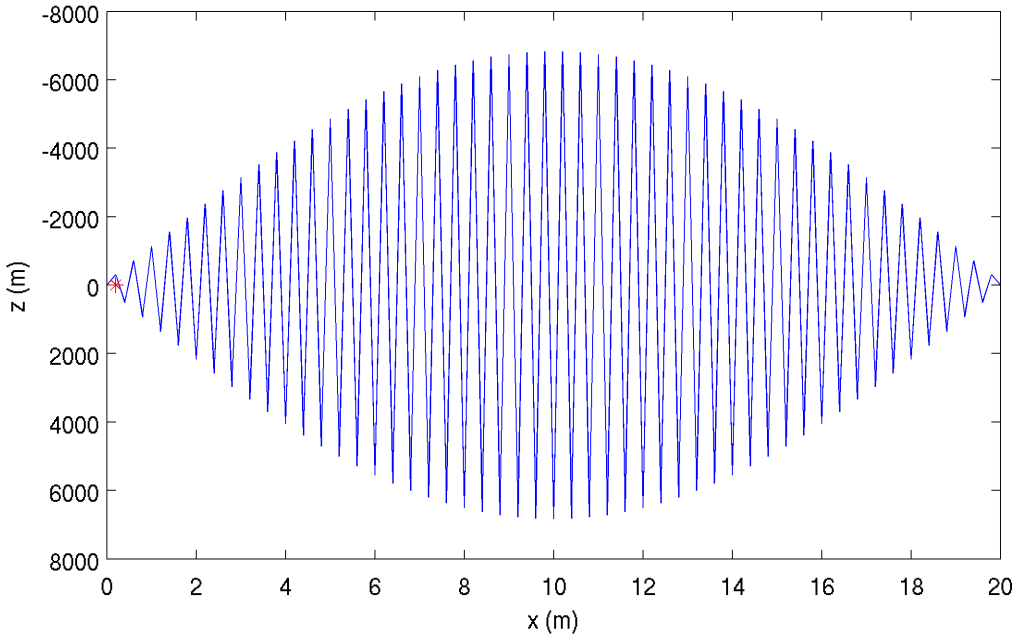

Figure 9: The displacement of the beam at time $t \approx 0.06$, using a CFL number slightly larger than $\frac{1}{4}$. The red star represents the position of the wheel.

This can numerically be verified by finding for multiple values of Δx (up to a certain precision) the lowest value of Δt so that the integration is unstable. Afterwards, one applies Richardson extrapolation to get the result.

The difference is significant. Although the time integration itself is of higher order, the time step Δt should be made roughly 16 times smaller when decreasing the mesh size by 2. For Euler Forward and Verlet integration this is only 4 times. This makes the RK4 method not preferable for many situations, as one easily needs an hour of computation time if more one wishes to have more than 50 discretisation points in the spatial direction.

6 Applying both local and global deformations

In Section 4 we discussed the physics involving a sphere falling on an elastic surface. Using Hertz theory we derived a differential equation that described the deformation of the elastic sphere. In Section 5 we described the total deformation and bending of a beam.

These two sections describe different phenomena. In Section 4 we only consider *local* deformations, the elastic surface itself does not move. Section 5 is focussed on the *global* deformation that occurs when there are forces being exerted on the beam.

In many dynamical contact problems, however, both phenomena occur at the same time. If a train is moving over a bridge, then the rails will be deformed slightly around each contact area. But the bridge as a whole will bend as well. These are two different systems but they are not completely separate of each other. The global deformation of the bridge will also effect the local deformation, since this changes the shape of the material around the initial contact point.

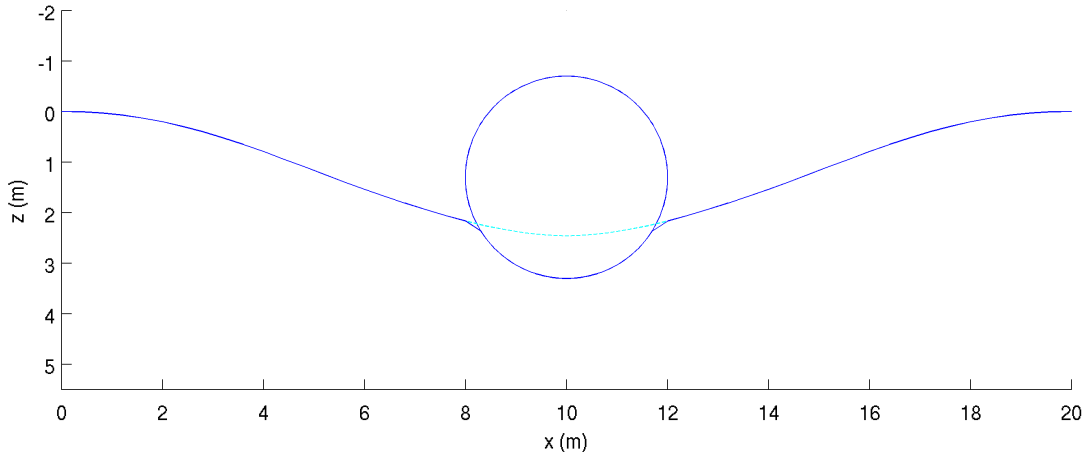

Figure 10: A visual interpretation of a train wheel on a bridge, combining global and local deformations. The cyan line represents the global deformation of the bridge if there is no local deformation. Note that the ratios in this figure are not realistic.

6.1 Applying the Finite Element Method

The straightforward approach that governs the global as well as the local deformations, is to use the Finite Element Method applied to equations (2.18). A simple mesh should be created for the beam. This should be either a 2D mesh or a 3D mesh, as long as it includes the thickness of the beam.

After the weak formulation is derived, the corresponding integrals should be computed (using sufficient basic functions). For the boundary elements the corresponding boundary conditions should be applied here. These boundary conditions depend on the time. Either Hertz theory or CONTACT should be used to determine the traction boundary conditions.

Another possibility is to use finite element software with built-in contact conditions.

Next, the time derivative $\frac{\partial u}{\partial t}$ can be computed for each element. An appropriate time integration scheme as in Section 3 should be used to integrate with respect to time.

The advantage of this method is that since the whole bridge (including its thickness) is discretised, both the global and local deformations are automatically included in the problem. This method results in an accurate solution of the problem if the elements are chosen small enough and can be used to verify the correctness of the real problem. It may be useful to use a grid which is finer around the contact region, so that the local deformation can be studied in much more detail.

6.2 Combining local and global deformations using CONTACT

The applied finite element method can become relatively costly if one wishes to compute deformations for very fine meshes, since it includes an extra dimension for the thickness of the beam. It is therefore an interesting question whether the results of Section 4 (about local deformations) and those of Section 5 (about global deformations) can somehow be used together to compute the solution accurately without it being too computationally expensive.

The main problem that will be researched during this thesis project is how CONTACT can be used used to compute the total deformation for these dynamical contact problems. Taking both local and global deformations into account, an algorithm should be developed that returns a good estimate of the total deformation, comparing this to the result of the Finite Element approach.

We will research how to do this more in detail in the following months. To give a slight outline; the total deformation will be computed iteratively at each time step. We start by computing the global deformation (not taking local deformation into account), using a given pressure p, and using this our initial guess of the total deformation. Then, the so called undeformed distance can be computed. This is the distance between the two bodies in a given coordinate system. CONTACT should then be called next. CONTACT then uses the undeformed distance to compute the local deformations in the contact area.

CONTACT also produces a pressure distribution around the initial contact point. This pressure should agree with the original pressure distribution we used to compute the global deformation. If this is not yet the case, the global deformation should be re-evaluated. This time, the newly computed pressure distribution should be used. This process is repeated until a stable solution is found, in which the pressure described in the local contact problem corresponds to the pressure described in the global problem. At this point, the global and local deformation are in agreement with each other and therefore the total deformation should be correct.

7 Research problems

The main goal of this research is to find a way to use CONTACT in dynamical contact problems, in particular those that occur in the rail industry.

It is often too hard to solve problems like these in full detail. Therefore, we will start looking at very simple (test) problems, solve these, and then gradually increase its complexity of the problem. These test problems are simple (often unrealistic) problems and are created solely to gain a better understanding of different parts of the main problem.

7.1 Deformation of an elastic half space

The first test program we will have a look at is the deformation of an elastic half space. We are interested in how its deformation can be computed if an object like a sphere exerts a force on the surface. A lot of analysis has already been done in Section 4. The total deformation is computed according to Hertz theory.

The first goal is to find out how CONTACT can be used to solve the same problem. Instead of using equation (4.18) to compute the normal force, we could also use CONTACT to do so. This will allow us to get a better understanding of CONTACT before we move on to the more difficult problems. With this approach, not only the total deformation is computed, but CONTACT also returns the pressure in the contact area and the displacement of the surface outside the contact area.

Applying the Finite Element method is also a good practice for this problem. Using Hertz theory we can derive boundary conditions for the elasticity equation (2.18). This can then be implemented using the Finite Element method to gain experience working with this. Furthermore, we will compare the result of the two different approaches.

7.2 Stationary situation of a wheel on a bridge

In the second problem we will research, we will take both global and local deformations into account. As before, consider an object such as a sphere or a (train) wheel on a bridge. For now, we will only look at the static problem. In this situation, the gravitational force of the wheel is the same as the normal force of the bridge. Furthermore, the wheel also stays in the same horizontal position.

For this problem we will try to compute the global and local deformation of the bridge. This should be done as described in Section 6. The main goal of this research problem is to solve the problem as described in Section 6.2, i.e. on how to derive the pressure distribution that corresponds to both the local and the global problem.

Applying the Finite Element method for this particular test problem is similar (even simpler, since there is no time dependency now) as for the previous test problem.

Note that if for this problem we use the parameters as in Table 2, the total global deformation will be extremely small (as in a total deformation of 0.2cm for a bridge of length 20m). This would have little to no effect on the local deformation and its pressure distribution in the contact area. It might therefore be advantageous to use different (unrealistic) parameters. For example, we could decrease the elasticity modulus by multiple factors. This will result in a much larger deformation of the bridge as a whole, and will therefore significantly change the undeformed distance between the bridge and the wheel.

7.3 Time dependent train / bridge simulation

The third and last research problem is an extension of the second problem. Instead of only looking at the stationary situation, we will also take time into account for this problem. We will consider a train that is moving as in Section 5.5, but now we will also take local deformation into account like in the second test problem.

Multiple time integration methods as described in Section 3 will be tested. This involves solving the pressure distribution problem (as in Problem 2 we just discussed). Furthermore, if an implicit integration scheme is used, an iterative solver should be applied to solve the implicit problem. Properties like accuracy, stability (and CFL number), computational complexity, and numerical damping and amplification will be researched and compared between all tested numerical integration schemes.

The results can then be further analysed. In particular, we are interested in what frequencies of the bridge will become dominant if a train moves over the bridge with a certain speed and whether resonance can occur.

References

[1] Landau, L. D., & Lifshitz, E. M. (1975). Elasticity theory. Pergamonn Press, second edition.

[2] Sadd, M. Fundamental equations of dynamic elasticity. Chapter 10. `http://personal.egr.uri.edu/sadd/mce565/Ch10.pdf`

[3] ESA. Glossary, `http://www.spaceflight.esa.int/impress/text/education/Glossary/Glossary_L.html`

[4] Popov, V. L. (Ed.). (2010). Contact mechanics and friction. Springer, Berlin.

[5] Geng, S. (1995). Construction of high-order symplectic PRK methods. J. Comput. Math, 13(1), 40-50.

[6] Verlet, L. (1967). Computer" experiments" on classical fluids. I. Thermodynamical properties of Lennard-Jones molecules. Physical review, 159(1), 98.

[7] Negrut, D., Rampalli, R., Ottarsson, G., & Sajdak, A. (2007). On an Implementation of the Hilber-Hughes-Taylor Method in the Context of Index 3 Differential-Algebraic Equations of Multibody Dynamics (DETC2005-85096). Journal of computational and nonlinear dynamics, 2(1), 73-85.

[8] Chung, J., & Hulbert, G. M. (1993). A time integration algorithm for structural dynamics with improved numerical dissipation: the generalized-α method. Journal of applied mechanics, 60(2), 371-375.

[9] Jerri, A. J. (1998). The Gibbs phenomenon in Fourier analysis, splines and wavelet approximations (Vol. 446). Springer Science & Business Media.

8 Appendix

The following matlab codes have been written and used for this report.

8.1 SphereOnPlane.m

This matlab code computes the deformation of the elastic sphere as discussed in Section 4. In this code, differential equation (4.23) is integrated using all of the integration schemes of Section 3. Picard iteration is used for those methods that are implicit.

```
% Settings:
g = 9.81; % Gravitational acceleration

% Material properties
E = 1.5E+8; % Young's modulus of Polybutadiene
nu = 0.50; % Poisson ratio
rho = .91E+6; % Density in g / m^3

% General settings:
R = 0.11; % Radius of ball
z0 = 1; % Height of ball at t=t0
v0 = 0; % Speed of ball at t=t0
eps = 0.0001; % Max picard error

% Time to integrate
dt = .002;
t0 = 0;
t1 = 1;



% Code starts here
t = t0:dt:t1;
m = 4/3*pi*R^3 * rho; % Mass of ball
n = length(t);
Estar = E / (2 * (1 - nu)^2);
z = [z0, zeros(1,n - 1)];
v = [v0, zeros(1,n - 1)];
a = [-g, zeros(1,n - 1)];
delta = [max(0,R - z0), zeros(1,n - 1)];


% z'' = A(z)
A = @(x) 4/3 * Estar * sqrt(R) * max(0, -x).^(3/2) / m - g;



if 0
% RK4
% Butcher tableau:
    % IIA Method
    T = [0,0,0,0;1/2,0,0,0;0,1/2,0,0;0,0,1,0];
    b = [1/6, 1/3, 1/3, 1/6];
```

```
    [kz,kv] = deal(zeros(4,1));
for i = 2:n
    kzo = kz;
    kvo = kv;
    for j = 1:4
        a = v(i - 1) + dt * T * kvo;
        kz(j) = a(j);
        a = A(z(i - 1) + dt * T * kzo);
        kv(j) = a(j);
    end
    z(i) = z(i - 1) + dt * b * kz;
    v(i) = v(i - 1) + dt * b * kv;
end

end




if 0
% RADAU5
% Butcher tableau:
if 1
    % IA Method
    T = [1/9, (-1-sqrt(6))/18, (-1+sqrt(6))/18;
         1/9, 11/45 + 7*sqrt(6)/360, 11/45 - 43*sqrt(6)/360;
         1/9, 11/45 + 43*sqrt(6)/360, 11/45 - 7*sqrt(6)/360];
    b = [1/9, 4/9 + sqrt(6)/36, 4/9 - sqrt(6)/36];
else
    % IIA Method
    T = [11/45 - 7*sqrt(6)/360, 37/225 - 169*sqrt(6)/1800, -2/225 + sqrt(6)/75;
      37/225 + 169*sqrt(6)/1800, 11/45 + 7*sqrt(6)/360, -2/225 - sqrt(6)/75;
      4/9 - sqrt(6)/36, 4/9 + sqrt(6)/36, 1/9];
    b = [4/9 - sqrt(6)/36, 4/9 + sqrt(6)/36, 1/9];
end
[kz, kv] = deal(zeros(3,1));
for i = 2:n
    % Integrating with RADAU5
    % Solving the system with Picard iteration
    while 1
        kzn = v(i - 1) + dt * T * kv;
        kvn = A(z(i - 1) + dt * T * kz);
        error = norm([kzn;kvn] - [kz;kv]);
        kz = kzn;
        kv = kvn;
        if error < eps
            z(i) = z(i - 1) + dt * b * kz;
            v(i) = v(i - 1) + dt * b * kv;
            break;
        end
    end
end

end

```

```
100
101
102 if 0
103 % Newmark
104 beta = 1/4;
105 gamma = 1/2;
106 for i = 2:n
107     % Integrating with HHT
108     % Solving the system with Picard iteration
109     z(i) = z(i - 1);
110     v(i) = v(i - 1);
111     a(i) = a(i - 1);
112     while 1
113         zn = z(i - 1) + dt * v(i - 1) + dt^2 / 2 * ((1 - 2*beta) * a(i - 1) + 2*beta * a(i));
114         vn = v(i - 1) + dt * ((1 - gamma) * a(i - 1) + gamma * a(i));
115         an = A(z(i));
116         if norm([zn - z(i); vn - v(i); an - a(i)]) < eps
117             break;
118         end
119         z(i) = zn;
120         v(i) = vn;
121         a(i) = an;
122     end
123 end
124 end
125
126
127
128
129 if 0
130 % HHT
131 alpha = -1/3;
132 beta = (1 - alpha)^2/4;
133 gamma = 1/2 - alpha;
134 A2 = @(zold,znew) -alpha * A(zold) + (1 + alpha ) * A(znew);
135
136 for i = 2:n
137     % Solving the system with Picard iteration
138     z(i) = z(i - 1);
139     v(i) = v(i - 1);
140     a(i) = a(i - 1);
141     while 1
142         zn = z(i - 1) + dt * v(i - 1) + dt^2 / 2 * ((1 - 2*beta) * a(i - 1) + 2*beta * a(i));
143         vn = v(i - 1) + dt * ((1 - gamma) * a(i - 1) + gamma * a(i));
144         an = A2(z(i - 1), z(i));
145         if norm([zn - z(i); vn - v(i); an - a(i)]) < eps
146             break;
147         end
148         z(i) = zn;
149         v(i) = vn;
150         a(i) = an;
151     end
152 end
153 end
154
```

```
155 if 0
156 % HHT 2
157 alpha = 0;
158 beta = 1/2 + alpha;
159 gamma = (1 + alpha)^2 / 4;
160 A2 = @(zold,znew) (1 - alpha ) * A(znew) + alpha * A(zold);
161
162 for i = 2:n
163     % Solving the system with Picard iteration
164     z(i) = z(i - 1);
165     v(i) = v(i - 1);
166     a(i) = a(i - 1);
167     while 1
168         zn = z(i - 1) + dt * v(i - 1) + dt^2 / 2 * ((1 - 2*beta) * a(i - 1) + 2*beta * a(i));
169         vn = v(i - 1) + dt * ((1 - gamma) * a(i - 1) + gamma * a(i));
170         an = A2(z(i - 1), z(i));
171         if norm([zn - z(i); vn - v(i); an - a(i)]) < eps
172             break;
173         end
174         z(i) = zn;
175         v(i) = vn;
176         a(i) = an;
177     end
178 end
179 end
180
181
182
183
184 if 0
185 % Generalized alpha
186 alphaM = 1;
187 alphaF = 1;
188 gamma = 1/2 - alphaM + alphaF;
189 beta = 1/4 * (1 - alphaM + alphaF)^2;
190
191 for i = 2:n
192     % Solving the system with Picard iteration
193     z(i) = z(i - 1);
194     v(i) = v(i - 1);
195     a(i) = a(i - 1);
196     while 1
197         zn = z(i - 1) + dt * v(i - 1) + dt^2 / 2 * ((1 - 2*beta) * a(i - 1) + 2*beta * a(i));
198         vn = v(i - 1) + dt * ((1 - gamma) * a(i - 1) + gamma * a(i));
199         an = (A((1 - alphaF) * z(i - 1) + alphaF * z(i)) - (1 - alphaM) * a(i - 1)) / alphaM;
200         if norm([zn - z(i); vn - v(i); an - a(i)]) < eps
201             break;
202         end
203         z(i) = zn;
204         v(i) = vn;
205         a(i) = an;
206     end
207 end
208 end
209
```

```
210
211
212
213 if 1
214 % Verlet
215
216 z(2) = z(1) + v(1) * dt + 1/2 * A(z(1)) * dt^2;
217 for i = 3:n
218     z(i) = 2 * z(i-1) - z(i-2) + A(z(i-1)) * dt^2;
219 end
220 % Computing the velocity
221 for i = 2:n-1
222     v(i) = (z(i + 1) - z(i - 1)) / (2 * dt);
223 end
224 v(n) = (z(n) - z(n - 1)) / dt;
225
226 end
227
228
229
230
231 if 0
232 % Euler Backward
233 for i = 2:n
234     % Solving the system with Picard iteration
235     z(i) = z(i - 1);
236     v(i) = v(i - 1);
237     while 1
238         delta(i) = max(0, -z(i));
239         a(i) = 4/3 * Estar * sqrt(R) * delta(i)^(3/2) / m - g;
240         zn = z(i - 1) + (t(i) - t(i-1)) * v(i);
241         vn = v(i - 1) + (t(i) - t(i-1)) * a(i);
242         if norm([zn - z(i); vn - v(i)]) < eps
243             break;
244         end
245         v(i) = vn;
246         z(i) = zn;
247     end
248 end
249 end
250
251
252
253 if 0
254 % Euler Forward
255 for i = 2:n
256     z(i) = z(i-1) + (t(i)-t(i-1)) * v(i-1);
257     v(i) = v(i-1) + (t(i)-t(i-1)) * a(i-1);
258
259     % Computing the acceleration
260     Fc = 4/3 * Estar * sqrt(R) * max(0, -z(i-1))^(3/2); % Contact force
261     Fz = -m*g;
262     a(i) = (Fc + Fz) / m;
263 end
264 end
```

```
265
266
267
268
269 plot(t,z);
270 xlabel('Time (s)');
271 ylabel('Height (m)');
272
273 figure;
274 Ekin = 1/2 * m * v.^2;
275 Egrav = m * g * z;
276 Eelas = 8/15 * Estar * sqrt(R) * max(0, -z).^(5/2);
277 Etotal = Ekin + Egrav + Eelas;
278 plot(t,Ekin,t,Egrav,t,Eelas,t,Etotal);
279 legend('Kinetic energy','Gravitational potential energy','Elastic potential energy','Total energy');
280 xlabel('Time (s)');
281 ylabel('Energy (J)');
282
283
284 if 0
285 f = figure;
286 xlabel('x (m)');
287 ylabel('z (m)');
288 j = 0:pi/100:2*pi;
289 spherex = R * cos(j);
290 spherey = R * sin(j) + R;
291 x = -2*R:.01:2*R;
292 for i = 1:n
293     plot(spherex, spherey + z(i));
294     hold on;
295     uz = max(0, -z(i)) - x.^2 / (2 * R);
296     plot(x, - max(0, uz));
297     axis([-R, R, min(z), max(z) + 2 * R]);
298     axis equal;
299     hold off;
300     pause(dt);
301 end
302 end
```

8.2 SphereCONTACT.m

In this code, the same problem is simulated. Instead of using equation (4.18), CONTACT is used to compute the contact force.

```
1  if 1
2  % Settings:
3  settings.g = 9.81; % Gravitational acceleration
4
5  % Material properties
6  settings.E = 1.5E+8; % Young's modulus of Polybutadiene
7  settings.nu = 0.50; % Poisson ratio
8  settings.G = settings.E / (2*(1 + settings.nu));
9
10 % General settings:
```

```
R = 0.11; % Radius of ball
rho = .91E+6; % Density in g / m^3
z0 = 1; % Height of ball at t=t0
v0 = 0; % Speed of ball at t=t0
eps = 0.0001; % Max picard error

% Time to integrate
dt = .005;
t0 = 0;
t1 = 1;

t = t0:dt:t1;
m = 4/3*pi*R^3 * rho;
n = length(t);
%Estar = E / (2 * (1 - nu)^2);

z = [z0; zeros(n - 1, 1)];
v = [v0; zeros(n - 1, 1)];
a = [-settings.g; zeros(n - 1, 1)];

% Creating the ball
settings.dx = .005;
settings.dy = .005;
x = -R:settings.dx:R;
y = -R:settings.dy:R;
[X,Y] = meshgrid(x,y);
Z = R - sqrt(R^2 - X.^2 - Y.^2);
ind = imag(Z) ~= 0;
Z(ind) = R;
%[X(ind), Y(ind)] = deal(NaN); % To prevent plotting
deformed = zeros([size(Z), n]);
Fg = m * settings.g;
Fn = zeros(n,1);

A = @(F) F / m - settings.g;


if 0
% Verlet
for k = 2:n
    [Fn(k),deformed(:,:,k)] = compute(Z + z(k - 1), settings);

    Fres = Fn(k) - Fg;
    a(k) = Fres / m;

    % Integrating using Verlet
    if k == 2
        z(2) = z(1) + v0 * dt + 1/2 * a(1) * dt^2;
    else
        z(k) = 2 * z(k - 1) - z(k - 2) + a(k - 1) * dt^2;
    end
end
end


```

```
66
67
68  if 0
69  % RADAU5
70  % Butcher tableau:
71  if 1
72      % IA Method
73      T = [1/9, (−1−sqrt(6))/18, (−1+sqrt(6))/18;
74           1/9, 11/45 + 7*sqrt(6)/360, 11/45 − 43*sqrt(6)/360;
75           1/9, 11/45 + 43*sqrt(6)/360, 11/45 − 7*sqrt(6)/360];
76      b = [1/9, 4/9 + sqrt(6)/36, 4/9 − sqrt(6)/36];
77  else
78      % IIA Method
79      T = [11/45 − 7*sqrt(6)/360, 37/225 − 169*sqrt(6)/1800, −2/225 + sqrt(6)/75;
80        37/225 + 169*sqrt(6)/1800, 11/45 + 7*sqrt(6)/360, −2/225 − sqrt(6)/75;
81        4/9 − sqrt(6)/36, 4/9 + sqrt(6)/36, 1/9];
82      b = [4/9 − sqrt(6)/36, 4/9 + sqrt(6)/36, 1/9];
83  end
84  [kz, kv, kzn, kvn] = deal(zeros(3,1));
85  for k = 2:n
86      % Integrating with RADAU5
87      % Solving the system with Picard iteration
88      while 1
89          for i = 1:3
90              kzn(i) = v(k − 1) + dt ∗ T(i,:) ∗ kv;
91              kvn(i) = A(compute(Z + z(k − 1) + dt ∗ T(i,:) ∗ kz, settings));
92          end
93          error = norm([kzn;kvn] − [kz;kv]);
94          kz = kzn;
95          kv = kvn;
96          if error < eps
97              z(k) = z(k − 1) + dt ∗ b ∗ kz;
98              v(k) = v(k − 1) + dt ∗ b ∗ kv;
99              [Fn(k),deformed(:,:,k)] = compute(Z + z(k), settings);
100             break;
101         end
102     end
103 end
104 end
105
106
107 if 1
108 % RK4
109 % Butcher tableau:
110     % IIA Method
111     T = [0,0,0,0;1/2,0,0,0;0,1/2,0,0;0,0,1,0];
112     b = [1/6, 1/3, 1/3, 1/6];
113     [kz,kv] = deal(zeros(4,1));
114 for k = 2:n
115     kzo = kz;
116     kvo = kv;
117     for i = 1:4
118         kz(i) = v(k − 1) + dt ∗ T(i,:) ∗ kvo;
119         kv(i) = A(compute(Z + z(k − 1) + dt ∗ T(i,:) ∗ kzo, settings));
120     end
```

```
121     z(k) = z(k - 1) + dt * b * kz;
122     v(k) = v(k - 1) + dt * b * kv;
123 end
124 end
125
126
127
128
129
130
131 end
132
133
134
135 if 1
136 f = figure;
137 xlabel('x (m)');
138 ylabel('y (m)');
139 zlabel('z (m)');
140 for k = 1:n
141     if z(k) < .2
142         surf(X, Y, Z + z(k)); % The ball (buttom)
143         hold on;
144         surf(X, Y, -Z + 2*R + z(k)); % The ball (top)
145         surf(X, Y, -deformed(:,:,k)) % The surface
146         axis equal;
147         axis([-R, R, -R, R, min(z), .1 + 2 * R]);
148         hold off;
149         pause(dt);
150     end
151 end
152 end
```

```
1  function [Fn, deformed] = compute(undeformed, settings)
2
3  if min(undeformed(:)) >= 0
4      Fn = 0;
5      deformed = zeros(size(undeformed));
6  else
7      % Computing the contact force
8
9      fid = fopen('temp.inp', 'w');
10     fprintf(fid,' 3 module %% result element 1, Contact patch 1\n');
11     fprintf(fid,'%% Next case 1\n');
12     fprintf(fid,' 200020 P-B-T-N-F-S PVTIME, BOUND , TANG , NORM , FORCE , STRESS\n');
13     fprintf(fid,' 022020 L-D-C-M-Z-E FRCLAW, DISCNS, INFLCF, MATER , RZNORM, EXRHS\n');
14     fprintf(fid,' 002111 G-I-A-O-W-R GAUSEI, IESTIM, MATFIL, OUTPUT, FLOW , RETURN\n');
15     fprintf(fid,' 200 30 30 1 1.0E-04 MAXGS , MAXIN , MAXNR , MAXOUT, EPS\n');
16     fprintf(fid,' 0.000 0.000 0.000 0.000 FUN, FUX, FUY, CPHI\n');
17     fprintf(fid,'%% Note: N=1 means FUN == FN, F=0 means FUX == CKSI, FUY == CETA\n');
18     fprintf(fid,' 0.3000 0.3000 FSTAT, FKIN\n');
19     fprintf(fid,sprintf(' %.4f %.4f %.4E %.4E SIGMA 1,2, GG 1,2\n',settings.nu,settings.nu,settings.G,1E
           +20));
20     fprintf(fid,' 1 IPOTCN\n');
```

```
21      fprintf(fid,sprintf(' %3d %3d %6.3f %6.2f %5.4f %5.4f MX,MY,XL,YL,DX,DY\n',size(undeformed,2),
            size(undeformed,1), -size(undeformed,2)/2 * settings.dx, -size(undeformed,1)/2 * settings.dy,
            settings.dx, settings.dy));
22      fprintf(fid,' 9 1 IBASE, IPLAN\n');
23      fprintf(fid,'%% PENETRATION, (1)-(2): SPECIFIED PER ELEMENT');
24
25      for i = 1:size(undeformed,1)
26          fprintf(fid,'\n');
27          for j = 1:ceil(size(undeformed,2) / 5)
28              fprintf(fid,'\n ');
29              p = undeformed(i,5*(j-1)+1:min(size(undeformed,2),5*(j-1)+5));
30              fprintf(fid,sprintf(' %.4E',p));
31          end
32      end
33
34      fprintf(fid,'\n%% UNRESTRICTED PLANFORM');
35      fprintf(fid,'\n 0 module');
36
37      tic;
38      system('../i01/contact_v13.1/bin/contact 2 temp.inp');
39
40      Fn = read_forces;
41      deformed = read_deformed_distance(undeformed);
42
43  end
44
45
46  end
```

```
1   function [Fn, Fx, Fy] = read_forces()
2
3   fid = fopen('temp.out');
4
5   while isempty(findstr(fgets(fid), 'TOTAL FORCES'))
6   end
7
8   fgets(fid);
9   array = str2num(fgets(fid));
10  Fn = array(1);
11  Fx = array(2);
12  Fy = array(3);
13
14  end
```

```
1   function p = read_penetration
2
3   A = load('temp.0001.mat','-ascii');
4
5   p = max(A(4:end, 8));
6
7   end
```

8.3 Beam.m

In this code, we consider a one dimensional beam with properties as can be found in Table 2. Differential equation (5.4) is discretised. The same time integration schemes are used to integrate with respect to time.

```
if 1
L = 20;
dx = L / 5;
dt = 0.053;
plotdt = 0.01;
t1 = 100;
eps = 1e-10;

E = 2E+10;
I = 0.0104;
L = 20;
rho = 7900;
mass = 76E+3;
veloc = 3;
g = 9.81;

x = (0:dx:L)';
t = (0:dt:t1)';
k = length(t);
m = length(x);
A = zeros(m);
[u,v] = deal(zeros(m,k));

% The following is based on the differential equation
% d^2 u / dt^2 = -d^4 u / dx^4 + p
% where d^4/dx^4 u_i = (u_(i-2) - 4u_(i-1) + 6u_i - 4u_(i+1) + u_(i+2)) / dx^4

% Different pressure distributions:
p = @(x,t) 0; % no force
%p = @(x,t) (x .* (1 - x))' * cos(t); % oscillating force
%p = @(x,t) point_force(x, veloc*t, mass*g) / rho; % moving point force
%p = @(x,t) point_force(x, veloc*3.3333, mass*g) / rho; % constant point force


% Creating the discretisation matrix
A(2, 1:4) = [4, -7, 4, -1] / dx^4; % u_0 is equal to u_2
for i = 3:m - 2
    A(i, i - 2 : i + 2) = [-1, 4, -6, 4, -1] / dx^4;
end
A(m - 1, m - 3:m) = [-1,4, -7, 4] / dx^4; % u_(m+1) is equal to u_(m-1)

A = A * E * I / rho;



% Initial condition:
u(:,1) = x.^2 .* (L - x).^2; % both u and d u / d x are 0 at x = 0 and x = 1
u(:,1) = 0; % zero everywhere
v(:,1) = 0;
```

```
50
51  if 1
52  % Integrating using Euler Forwards
53  for j = 2:k
54      u(:,j) = u(:,j - 1) + dt * v(:,j - 1); % du/dt = v
55      v(:,j) = v(:,j - 1) + dt * (A * u(:,j)); % dv/dt = d^2u/dt^2 = -d^4/dx^4 u + p
56  end
57  end
58
59  if 0
60  % Euler Backwards
61  B = eye(2*m) - dt * [zeros(m), eye(m); A, zeros(m)];
62  % Computing the inverse of B:
63  Binv = B^(-1);
64
65  for j = 2:k
66      z = Binv * [u(:, j - 1); v(:, j - 1)];
67      u(:, j) = z(1:m);
68      v(:, j) = z(m+1:end);
69  end
70  end
71
72  if 0
73  % Euler Backwards old w/picard
74  for j = 2:k
75      % Solving the system with Picard iteration
76      un = u(:, j - 1);
77      vn = v(:, j - 1);
78      iter = 1;
79      while 1
80          u(:, j) = u(:, j - 1) + dt * vn;
81          v(:, j) = v(:, j - 1) + dt * (A * un);
82          if norm([un - u(:, j); vn - v(:, j)]) < eps
83              break;
84          end
85          un = u(:, j);
86          vn = v(:, j);
87          iter = iter + 1;
88          if iter >= 100
89              disp('Picard iteration does not converge.');
90              return;
91          end
92      end
93  end
94  end
95
96
97  if 0
98  % Verlet
99
100 u(:,2) = u(:,1) + v(:,1) * dt + 1/2 * dt^2 * (A*u(:,1) + p(x,0));
101 for i = 3:k
102     u(:,i) = 2 * u(:,i - 1) - u(:,i - 2) + dt^2 * (A*u(:,i - 1) + p(x, (i - 2) * dt));
103 end
104 % Computing the velocity
```

```
105 for i = 2:m−1
106     v(:,i) = (u(:,i + 1) − u(:,i − 1)) / (2 ∗ dt);
107 end
108 v(:,m) = (u(:,m) − u(:,m − 1)) / dt;
109
110 end
111
112
113
114 if 0
115 % RK4
116 f = @(z,t) A∗z + p(x, t);
117 eps = 1e−10;
118 % Butcher tableau:
119
120 T = [0,0,0,0;1/2,0,0,0;0,1/2,0,0;0,0,1,0];
121 b = [1/6, 1/3, 1/3, 1/6];
122
123 [ku,kv] = deal(zeros(m,4));
124 for i = 2:k
125     kuo = u(:,j − 1);
126     kvo = kv;
127     for j = 1:4
128         ku(:,j) = v(:,i − 1) + dt ∗ ku ∗ T(:,j);
129         kv(:,j) = f(u(:,i − 1) + dt ∗ kv ∗ T(:,j), (i − 1)∗dt);
130     end
131     u(:,i) = u(:,i − 1) + dt ∗ ku ∗ b';
132     v(:,i) = v(:,i − 1) + dt ∗ kv ∗ b';
133 end
134
135 end
136
137
138
139
140 if 0
141 % RADAU5
142 f = @(z,t) A∗z + p(x, t);
143 eps = 1e−10;
144 % Butcher tableau:
145 if 1
146     % IA Method
147     T = [1/9, (−1−sqrt(6))/18, (−1+sqrt(6))/18;
148           1/9, 11/45 + 7∗sqrt(6)/360, 11/45 − 43∗sqrt(6)/360;
149           1/9, 11/45 + 43∗sqrt(6)/360, 11/45 − 7∗sqrt(6)/360];
150     b = [1/9, 4/9 + sqrt(6)/36, 4/9 − sqrt(6)/36];
151 else
152     % IIA Method
153     T = [11/45 − 7∗sqrt(6)/360, 37/225 − 169∗sqrt(6)/1800, −2/225 + sqrt(6)/75;
154       37/225 + 169∗sqrt(6)/1800, 11/45 + 7∗sqrt(6)/360, −2/225 − sqrt(6)/75;
155       4/9 − sqrt(6)/36, 4/9 + sqrt(6)/36, 1/9];
156     b = [4/9 − sqrt(6)/36, 4/9 + sqrt(6)/36, 1/9];
157 end
158 [ku, kv, kun, kvn] = deal(zeros(m,3));
159
```

```
for i = 2:k
    % Integrating with RADAU5
    % Solving the system with Picard iteration
    iter = 0;
    while 1
        for j = 1:3
            kun(:,j) = v(:,i - 1) + dt * ku * T(:,j);
            kvn(:,j) = f(u(:,i - 1) + dt * kv * T(:,j), (i - 1)*dt);
        end
        error = norm([kun(:);kvn(:)] - [ku(:);kv(:)]);
        nrm = norm([kun(:);kvn(:)]);
        if iter > 100
            break
        end
        ku = kun;
        kv = kvn;
        if error < eps || error/nrm < eps
            u(:,i) = u(:,i - 1) + dt * ku * b';
            v(:,i) = v(:,i - 1) + dt * kv * b';
            break;
        end
        iter = iter + 1;
    end
        if iter > 100
            break;
        end
end
end


end

if 1
f = figure;
xlabel('x (m)');
ylabel('z (m)');
set(gca, 'ydir','reverse');
for i = 1:plotdt/dt:k
    plot(x,u(:,round(i)));
    hold on;
    axis([0,L,min(u(:)),max(u(:))]);
    set(gca, 'ydir','reverse');
    hold off;
    pause(0.001);
end
end
```