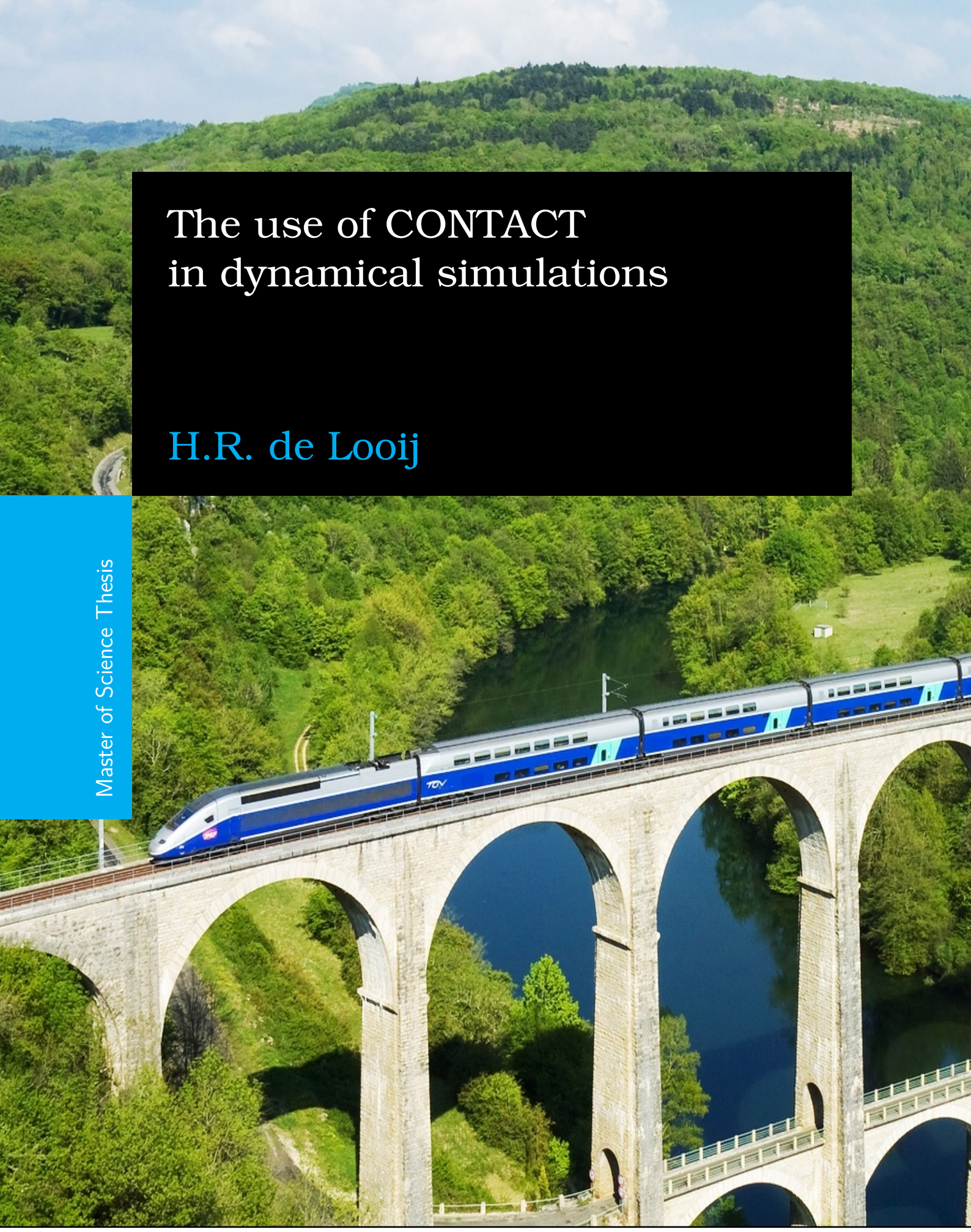
The use of CONTACT in dynamical simulations

H.R. de Looij

Master of Science Thesis



TU Delft Delft University of Technology

VORTECH

MSc thesis APPLIED MATHEMATICS

The use of CONTACT in dynamical simulations

Hugo de Looij

Delft University of Technology

Supervisors

Dr.ir. E.A.H. Vollebregt	VORtech
Prof. dr. ir. C. Vuik	TU Delft

Other members of the graduation committee

Dr. P. Wilders	TU Delft

November, 2015	Delft

Contents

1 Introduction **1**

2 The research problem **3**
2.1 Literature for dynamical contact problems 3
2.2 Deformation of an elastic half-space 4
2.3 Global deformation of a bridge 4
2.4 Full train-bridge simulation 5

3 Introduction to linear elasticity theory **7**
3.1 The elasticity components 7
3.1.1 Displacements 7
3.1.2 Strain 8
3.1.3 Stress 8
3.2 Relation between the elasticity components 9
3.2.1 Strain-displacement relations 9
3.2.2 Stress-strain relations 9
3.3 Other elastic properties 11
3.4 The linear elasticity equations 11
3.5 The boundary value problem 12

4 The basics of contact mechanics and Hertz theory **13**
4.1 Deformation of an elastic half space under stress 14
4.2 Contact between a rigid sphere and an elastic surface 15
4.3 Contact between two curved surfaces 16
4.4 Contact between elastic bodies 16
4.5 Using CONTACT to compute p and F 17
4.5.1 The penetration and undeformed distance 17
4.5.2 Using CONTACT 18
4.5.3 Example input file 19
4.6 Comparison between Hertz theory and CONTACT 21

5 Numerical methods for contact problems **23**
5.1 Newton's equations of motion for contact problems 23
5.2 Forward / Backward Euler 24
5.3 Runge Kutta / Radau methods 24
5.4 The Verlet method 26
5.5 Leapfrog integration 26
5.6 Newmark's method 27
5.7 The HHT method 27

5.8 The generalized-α method . . . 28
5.9 Adams methods . . . 28
5.10 Backward differentiation formulas . . . 29

6 Rigid body motion **31**
6.1 Derivation of the differential equation . . . 32
6.2 Properties of the solution . . . 32
6.3 Solving the problem numerically . . . 33
6.4 Using CONTACT to solve the problem . . . 36
6.5 Conclusion . . . 38

7 Discretisation of the elasiticty equations **41**
7.1 Formulation of the continuous problem . . . 42
7.1.1 The problem . . . 42
7.1.2 Computation of the top boundary condition using Hertz theory . . . 43
7.1.3 Formulation of the differential algebraic equation . . . 44
7.2 Discretisation . . . 45
7.2.1 Implementation with the Forward Euler approach . . . 48
7.2.2 Implementation with the Backward Euler approach . . . 49
7.3 Dynamic boundary conditions . . . 49
7.4 Numerical results . . . 50
7.5 Conclusion . . . 51

8 Global deformations of a bridge **55**
8.1 The 1D dynamic beam equation . . . 55
8.2 Boundary and initial conditions . . . 56
8.3 The discrete problem . . . 57
8.3.1 Discretisation of the differential equation . . . 57
8.3.2 Implementation of the boundary conditions . . . 58
8.3.3 The CFL condition . . . 59
8.3.4 Implicit methods . . . 60
8.4 Modal analysis . . . 61
8.4.1 Mode shapes . . . 61
8.4.2 Boundary conditions . . . 62
8.4.3 Orthogonality of the mode shapes . . . 63
8.4.4 Modal analysis using the discretisation . . . 65
8.5 The stationary case . . . 68
8.5.1 The analytic solution . . . 68
8.5.2 Numerical approximation using mode shapes . . . 69
8.5.3 Error analysis . . . 69
8.5.4 Numerical validation . . . 71
8.6 Forced vibrations . . . 71
8.6.1 The differential equation . . . 71
8.6.2 Numerical approximation using mode shapes . . . 73
8.7 A simple moving load problem . . . 74
8.7.1 The discrete problem . . . 75
8.7.2 The modal approach . . . 76
8.7.3 Resonance . . . 77
8.8 Conclusion . . . 78

9 Combining local and global deformations **81**

9.1 Assumptions . . . 81
9.1.1 The total deformation of a beam . . . 81
9.1.2 The penetration and approach for a globally deformed beam . . . 82
9.1.3 The pressure for a globally deformed beam . . . 82
9.1.4 The quasi-static local deformation . . . 82
9.2 Combining global and local deformations . . . 83
9.3 The stationary problem . . . 83
9.4 The time-dependent problem . . . 85
9.4.1 Explicit integration schemes . . . 86
9.4.2 Implicit integration schemes . . . 87

10 Full train-bridge simulation **89**
10.1 Formulation of the problem . . . 89
10.2 The system of differential equations . . . 91
10.3 Implementation of Backward Euler . . . 91
10.4 The Picard approach . . . 93
10.4.1 The algorithm . . . 93
10.4.2 Numerical results . . . 95
10.5 The Quasi-Newton approach . . . 96
10.5.1 Linearisation . . . 96
10.5.2 Applied to Backward Euler . . . 99
10.5.3 Solving the linear system . . . 101
10.6 The algorithm . . . 103
10.7 Numerical results . . . 105
10.7.1 Visual observations . . . 105
10.7.2 Effect on the approach and dynamic amplification factor . . . 107
10.7.3 Numerical convergence . . . 109

11 Summary and Further Research **111**
11.1 Summary and conclusions . . . 111
11.2 Further research . . . 113

A Matlab codes **117**
A.1 run_contact.m . . . 117
A.2 SphereOnPlaneCONTACT.m . . . 119
A.3 ElasitictySphereHalfplane.m . . . 122
A.4 StationaryBeam.m . . . 128
A.5 FullBridgeModes.m . . . 130

Chapter 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. Consider a train moving over a track. Multiple forces being exerted on the rails, with gravity being the most important one. The forces will result 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, and even estimate the probability of train derailment. 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 Prof.dr.ir Joost Kalker of the Delft University of Technology. In 2000, VORtech has taken over the software. It is now being further developed by Dr.ir. Edwin Vollebregt, who has been my supervisor for this Master project.

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

In this thesis report, the physics and mathematical theory behind dynamical contact problems, i.e. time-dependent contact problems, is discussed. An algorithm capable of simulating the deformation of a bridge and the wheels of a train that is moving over it will be developed. The main goal is that this algorithm will be much less computationally expensive then the usual finite element models. To achieve this, CONTACT plays a crucial role.

Chapter 2

The research problem

The main research goal for the project is to understand how CONTACT can be used in combination with time integration schemes for dynamical contact problems, in particular those that occur in the rail industry. Specifically, the main research problem we will be looking at involves a train moving over a bridge. How does the bridge and the train wheels deform as the result of this and how can this deformation be computed efficiently?

Because of the complexity of problems involving contact dynamics, it is often too hard to solve them in full detail. For instance, phenomena such as friction, slip, and adhesion can occur in contact problems. These phenomena are hard to describe in full detail, so assumptions will be made to simplify the problem. Throughout this thesis, as an example, we will ignore the effects of friction.

We will start looking at very simple test problems and then solve these. Afterwards, the complexity of these problem will be gradually increased. These test problems are simple (often unrealistic) problems and are created solely to gain a better understanding of different parts of the main problem.

The thesis can roughly be split in four different parts. Each part discusses a separate topic, each of them being required to solve the main research problem. The research topic and goal of each part will be outlined here.

2.1 Literature for dynamical contact problems

The first part, which consists of Chapters 3 to 5, contains necessary literature for dynamical contact problems. In Chapter 3, we will describe the basics of linear elasticity theory. This is used to derive a differential equation that describes the deformation of a single elastic object. Chapter 4 is about contact mechanics, which describes the physics and boundary conditions of two contacting objects. Both Hertz theory as well as CONTACT are used as contact model. Time integration schemes for dynamical contact problems are discussed in Chapter 5. These are used to solve multiple differential equations occurring in contact mechanics which will be derived throughout the thesis.

2.2 Deformation of an elastic half-space

The second part of this thesis is about local deformations and consists of Chapters 6 and 7. Here, we discuss the deformation of objects that can only deform locally as the result of compression of the material.

A simple problem involving the deformation of an elastic half-space will be discussed. We are interested in how this local deformation can be computed as function of time if an object like a sphere is dropped on the surface. We will show in Chapter 6 how Hertz theory can be used to derive a simple one dimensional differential equation for the rigid height of the ball. Alternately, CONTACT can be used as replacement for Hertz theory. This is applicable for more general situations and returns additional information which will be crucial for the development of the eventual algorithm.

Additionally, in Chapter 6 we will solve the differential equation describing the falling sphere by using the time integration schemes as discussed in Chapter 5. Since the differential equation is simple, this test problem can easily be used to validate the numerical time integration schemes and to get a basic idea of the pros and cons of each scheme. Specifically, we are interested in the stability, accuracy and energy conservation of each numerical scheme. This problem will also be used to gain more understanding on how to run CONTACT and how its output can be used.

The problem becomes more complex when one is interested in the total deformation of the half space as function of x and y. We will discuss two ways of computing this deformation. The first one makes use of CONTACT which solves the quasi-static elasticity equations. The advantage of this approach is that this deformation is easily computed. However, inertia is completely ignored; hence the result is not completely realistic.

For a more accurate approximation, we will discretise the elasticity equations for the half-plane and combine this with the correct boundary conditions and a good time integration scheme. This will be thoroughly discussed in Chapter 7. This approach, however, is computationally very expensive.

2.3 Global deformation of a bridge

In Chapter 8, the third part of this thesis, we will consider global deformations. Global deformation represents the deformation of an object (in this case, a bridge) that occurs by the displacement of the objects on a global scale. The main difference between an elastic bridge and an elastic half-plane, is that a bridge is unsupported at the bottom. The bridge will therefore deform *globally*, while a half-plane can only deform by compression the material. Both phenomena have different properties and will hence be discussed separately.

The test problem we will discuss in this chapter is a bridge that deforms globally as a result of a given pressure distribution that is being exerted on the bridge. Local deformation is ignored for now. First, we will show how to compute the stationary solution. Next, time will be taken into account; this will be used to solve a simple moving-load problem. The time-dependent deformation of a bridge as the result of a given pressure distribution (representing a moving train, for example) will then be solved.

The bridge will be modelled as a long thin beam. Its deformation will be approximated by solving the 1D Euler-Bernoulli beam equation. We will mainly focus on a modal approach of

doing so; the solution will be constructed by using so-called mode shapes. We will show how these functions can be computed or approximated and how they can be used to approximate the global deformation. A system of independent ordinary differential equations will be derived and solved.

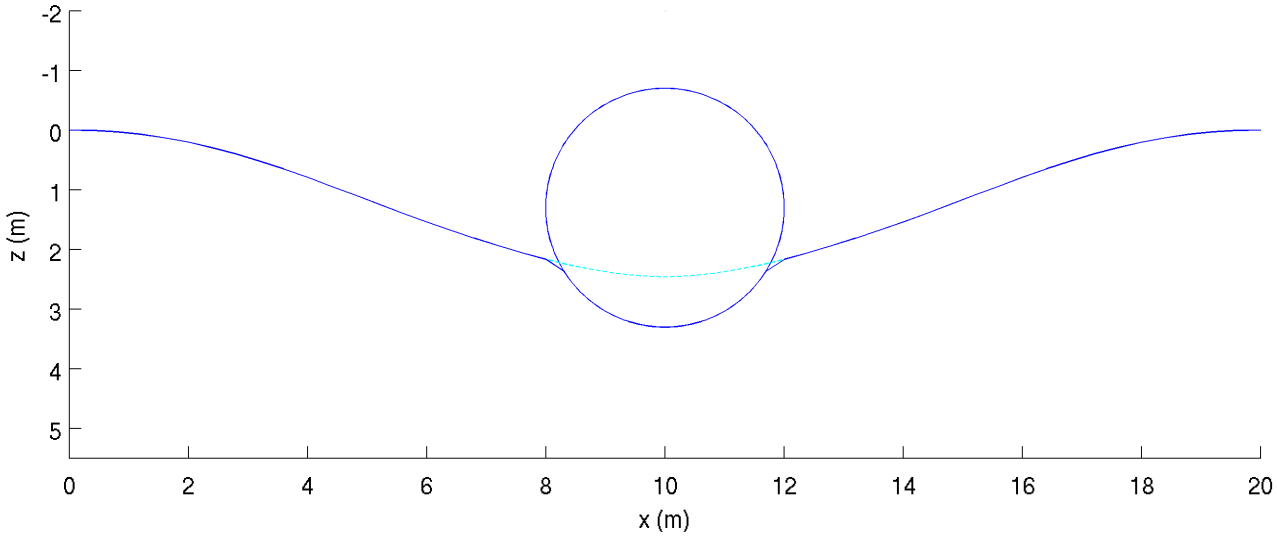


Figure 2.1: Representation of a bridge that is deformed both globally as well as locally.

2.4 Full train-bridge simulation

The last part of thesis consists of Chapter 9 and 10. Here, we will combine the previously discussed theory in order to perform a simulation of train-bridge contact.

If a train wheel exerts a force on the bridge, the bridge will deform as a whole (i.e. globally). In reality, however, the bridge will also behave similarly to the half-plane; a 'gap' will appear around the contact area that contains a part of the wheel. A visual (unrealistic) representation can be seen in Figure 2.1.

In this last part of thesis we will focus on the interaction between the global and local deformation. Our goal will be develop an algorithm to solve the main problem for this project, i.e. how to compute the total deformation of a bridge accurately (without being too computationally expensive) for a train moving over it. This combines the theory of global and local deformations and CONTACT plays a central role in this algorithm.

In Chapter 9 we will discuss the interaction between the global and local deformation of the bridge. The total deformation will be modelled as the sum of the global and the local deformation. The problem that occurs here is that the global and local deformation are the result of two separate problems which are not independent of each other. We will propose and test algorithms for slightly simplified models. For instance, we will first solve the stationary problem.

Chapter 10 combines all the previously discussed theory in order to perform a full train-bridge simulation. This involves the implementation of the integration schemes and the iterative solver. We will counter several stability problems. To solve this, we will propose an improvement of the iterative solver which will both increase stability as well as the convergence rate without a big increase in the total amount of required computational power.

Chapter 3

Introduction to linear elasticity theory

Contact mechanics is the study that deals with the physics of two contacting 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.

Before getting into the contact between two different objects, we will consider a single object being deformed. This deformation is described by elasticity theory. In this chapter, we will focus on the dynamic deformation of a single general three dimensional elastic object. The Cartesian coordinate system is used and a vector $\mathbf{u}$ can be denoted both as $\mathbf{u} = (u_x, u_y, u_z)$ as well as $\mathbf{u} = (u_1, u_2, u_3)$. Einstein convention is used frequently.

3.1 The elasticity components

In elasticity theory, there are three different important elasticity components: the displacements, the strain, as well as the stress. We will first describe each of these components separately and then get into the relations between each component in order to derive a differential equation for the dynamic deformation of an object.

3.1.1 Displacements

The displacement is the variable in elasticity theory we are generally interested in. Often, one is interested in determining the change of shape of an object such as a bridge or wheel. The displacement describes the change of shape of an elastic body as function of the time.

Consider an undeformed object in rest and focus one particle in this object. Now consider a force at the boundary of the object pushing in any direction. This can cause the object to be stretched or compressed. In particular, the particle we have chosen can move in any direction. 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.

3.1.2 Strain

Strain represents the stretching (or compressing) of an object. Once again, focus on one particle in 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 change. Strain is dimensionless quantity and represents the relative change of the position of points in the body.

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 correspond 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 (often notated by γ) represent the change of angle between two points as their distance tends to zero. These shear strains are always symmetric, so for example $\gamma_{xy} = \gamma_{yx}$.

The strain tensor ε is a matrix that contains the nine strain components. Due to the symmetry of the shear strains, the matrix is symmetric. It is defined by

$$\varepsilon = \begin{pmatrix} \varepsilon_{xx} & \gamma_{xy} & \gamma_{xz} \\ \gamma_{yx} & \varepsilon_{yy} & \gamma_{yz} \\ \gamma_{zx} & \gamma_{zy} & \varepsilon_{zz} \end{pmatrix} \tag{3.1}$$

3.1.3 Stress

Stress is the physical quantity that represents the force per unit area that is being exerted on a particle in a body. The unit of stress is Nm^{-2}. If there is stress, it means that the material is under tension or compression. 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 shear stresses (often denoted as τ). Similarly for the strain, the shear stresses are symmetric; this is a result of the conservation of angular momentum. We can define a stress tensor σ as

$$\sigma = \begin{pmatrix} \sigma_{xx} & \tau_{xy} & \tau_{xz} \\ \tau_{yx} & \sigma_{yy} & \tau_{yz} \\ \tau_{zx} & \tau_{zy} & \sigma_{zz} \end{pmatrix} \tag{3.2}$$

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

$$\mathbf{r} = \sigma \mathbf{n} \tag{3.3}$$

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

3.2 Relation between the elasticity components

3.2.1 Strain-displacement relations

By definition of the strains, the longitudinal strains are simply the spatial derivative of the displacement in the same direction. The strain components are therefore connected to the displacements by the following equations

$$\begin{aligned}
\varepsilon_{xx} &= \frac{\partial u_x}{\partial x} \\
\varepsilon_{yy} &= \frac{\partial u_y}{\partial y} \\
\varepsilon_{zz} &= \frac{\partial u_z}{\partial z} \\
\gamma_{xy} &= \frac{\partial u_x}{\partial y} + \frac{\partial u_y}{\partial x} \\
\gamma_{xz} &= \frac{\partial u_x}{\partial z} + \frac{\partial u_z}{\partial x} \\
\gamma_{yz} &= \frac{\partial u_y}{\partial z} + \frac{\partial u_x}{\partial z}
\end{aligned} \tag{3.4}$$

We now redefine the strain components in smart way such that system (8.38) can be written in a more compact way. We define

$$e_{ij} = \begin{cases} \varepsilon_{ii} & \text{if } i = j \\ \frac{1}{2}\gamma_{ij} & \text{if } i \neq j \end{cases} \tag{3.5}$$

So that (8.38) can be rewritten as

$$e_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) \qquad \text{for all } 1 \leq i, j \leq 3 \tag{3.6}$$

or even shorter

$$e = \frac{1}{2}\left(\frac{\mathrm{d}\mathbf{u}}{\mathrm{d}\mathbf{x}} + \left(\frac{\mathrm{d}\mathbf{u}}{\mathrm{d}\mathbf{x}}\right)^T\right) \tag{3.7}$$

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

3.2.2 Stress-strain relations

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.

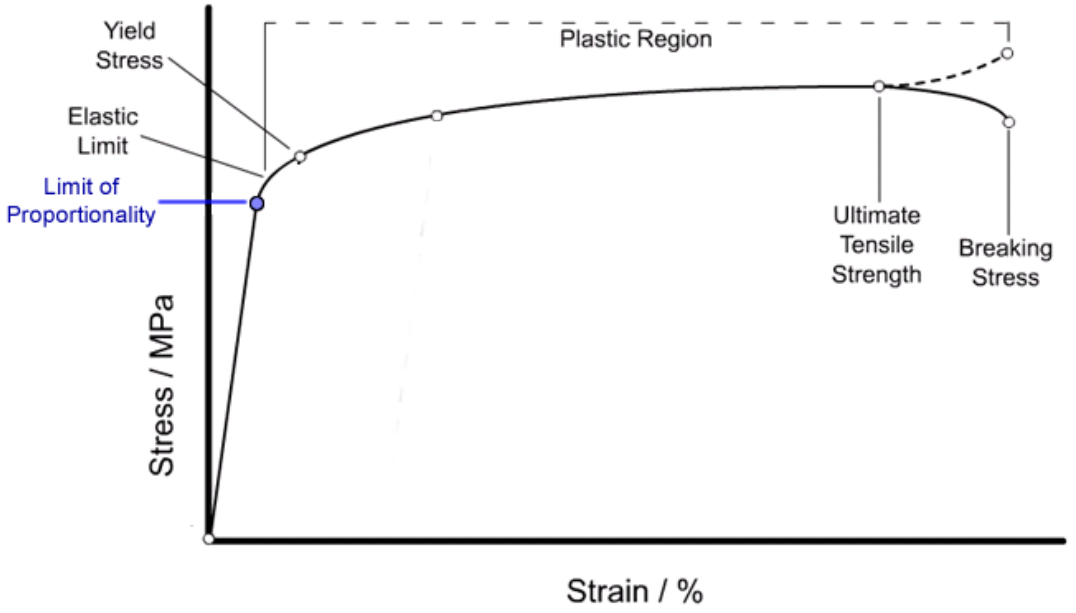


Figure 3.1: The relation between strain and stress. Source: [3].

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 3.1. If the strain is larger than this value, the object can start to deform plastically (i.e. the deformation will become permanent) or even break down. Throughout this paper, we assume that the strains do not surpass the limit of proportionality. This assumption that the ratio between the stress and the strain is linear, is the principle of linear elasticity theory.

According to Hooke's law, we have the relation [2]

$$\begin{aligned}
\sigma_{xx} &= \lambda(e_{xx} + e_{yy} + e_{zz}) + 2Ge_{xx} \\
\sigma_{yy} &= \lambda(e_{xx} + e_{yy} + e_{zz}) + 2Ge_{yy} \\
\sigma_{zz} &= \lambda(e_{xx} + e_{yy} + e_{zz}) + 2Ge_{zz} \\
\sigma_{xy} &= Ge_{xy} \\
\sigma_{xz} &= Ge_{xz} \\
\sigma_{yz} &= Ge_{yz}
\end{aligned} \tag{3.8}$$

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

$$\sigma_{ij} = \lambda e_{kk}\delta_{ij} + 2Ge_{ij} \quad \text{for } 1 \leq i, j \leq 3 \tag{3.9}$$

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

$$\sigma = \lambda \mathrm{Tr}(e)I + 2Ge \tag{3.10}$$

3.3 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, these variables are dependent of each other. 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 ν.

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

$$E = \frac{G(3\lambda + 2G)}{\lambda + G} \tag{3.11}$$

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

$$\nu = \frac{\lambda}{2(\lambda + G)} \tag{3.12}$$

3.4 The linear elasticity equations

Using the relations between stress and displacement as well as the relations between stress and strain, the linear elasticity equations can be derived. This set of equations is based on the equations of motion. According to Newton's equation of motion, the force in a particular direction at each particle 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 the same direction. It can be shown that the corresponding equations of motions are [2]

$$\begin{aligned}
\frac{\partial \sigma_{xx}}{\partial x} + \frac{\partial \sigma_{xy}}{\partial y} + \frac{\partial \sigma_{xz}}{\partial z} + F_x &= \rho \frac{\partial^2 u_x}{\partial t^2} \\
\frac{\partial \sigma_{yx}}{\partial x} + \frac{\partial \sigma_{yy}}{\partial y} + \frac{\partial \sigma_{yz}}{\partial z} + F_y &= \rho \frac{\partial^2 u_y}{\partial t^2} \\
\frac{\partial \sigma_{zx}}{\partial x} + \frac{\partial \sigma_{zy}}{\partial y} + \frac{\partial \sigma_{zz}}{\partial z} + F_z &= \rho \frac{\partial^2 u_z}{\partial t^2}
\end{aligned} \tag{3.13}$$

Here, F_x, F_y, and F_z are body forces. Gravity is an example of a body force in the z direction. The partial derivatives $\frac{\partial \sigma_{ij}}{\partial x_j}$ in (3.13) can be seen as the internal forces caused by the stresses in the object.

Equation (3.13) can be simplified to

$$\frac{\partial \sigma_{ij}}{\partial x_j} + F_i = \rho \frac{\partial^2 u_i}{\partial t^2} \tag{3.14}$$

or by using tensor notation

$$\nabla \cdot \sigma + \mathbf{F} = \rho \frac{\partial^2 \mathbf{u}}{\partial t^2} \tag{3.15}$$

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

It is possible to eliminate σ from (3.13) and derive a differential equation using only the displacement variables. To do so, substitute (8.38) in (3.8). Next, substitute the resulting expressions for σ in (3.13). This results in the alternative equations of motion

$$\begin{aligned}
G\Delta u + (\lambda + G)\frac{\partial}{\partial x}\left(\frac{\partial u_x}{\partial x} + \frac{\partial u_y}{\partial y} + \frac{\partial u_z}{\partial z}\right) + F_x &= \rho \frac{\partial^2 u_x}{\partial t^2} \\
G\Delta v + (\lambda + G)\frac{\partial}{\partial y}\left(\frac{\partial u_x}{\partial x} + \frac{\partial u_y}{\partial y} + \frac{\partial u_z}{\partial z}\right) + F_y &= \rho \frac{\partial^2 u_y}{\partial t^2} \\
G\Delta w + (\lambda + G)\frac{\partial}{\partial z}\left(\frac{\partial u_x}{\partial x} + \frac{\partial u_y}{\partial y} + \frac{\partial u_z}{\partial z}\right) + F_z &= \rho \frac{\partial^2 u_z}{\partial t^2}
\end{aligned} \tag{3.16}$$

or alternatively

$$G\frac{\partial^2 u_i}{\partial x_j^2} + (\lambda + G)\frac{\partial^2 u_j}{\partial x_i x_j} + F_i = \rho \frac{\partial^2 u_i}{\partial t^2} \qquad \text{for } 1 \leq i \leq 3 \tag{3.17}$$

or in vector notation as

$$G\Delta \mathbf{u} + (\lambda + G)\nabla(\nabla \cdot \mathbf{u}) + \mathbf{F} = \rho \frac{\partial^2 \mathbf{u}}{\partial t^2} \tag{3.18}$$

3.5 The boundary value problem

Although (3.16) seems easier to use than (3.13) since we eliminated the use of the stress variable, it is less practical. In contact mechanics, boundary conditions for the elasticity equations are crucial. There are two main boundary conditions, conditions for the traction and conditions for the displacement at the surface. Both can be used at the same time for different boundaries.

Traction boundary conditions should be applied on a boundary if a known 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 $-\mathbf{p}_0 = \mathbf{r} = \sigma \cdot \mathbf{n}$ 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 (3.16) is of second order with respect to the time variable, both initial displacements $\mathbf{u}$ as well as the velocity for the displacements $\dot{\mathbf{u}}$ should be specified. Usually, we suppose that the object is in rest at $t = 0$, which corresponds to $\mathbf{u} = \dot{\mathbf{u}} = 0$ as the initial conditions.

In Chapter 7, we will create a finite difference discretisation of the elasticity equations. Formulation (3.13) is used, since we will describe boundary conditions for the traction.

Chapter 4

The basics of contact mechanics and Hertz theory

In the previous chapter, the deformation of a single object has been discussed. The elasticity equations (3.13) describe the change of shape of the object under stress. For systems with multiple bodies, however, the situation is more complex. Two bodies can interact with each other and cause deformation. Although the elasticity equations (3.13) are still relevant for all bodies, the boundary conditions are not typically known in advance. Contact mechanics involves the computation of the pressure distribution at the boundary of two touching objects. It can also involve friction and similar phenomenons, but we will usually neglect this.

The Hertz model, developed in 1880, is the first contact model which describes contact mechanics accurately. This theory describes the contact between two elastic spheres or between a sphere and a half-space. The pressure at the boundary as well as the contact force is described in these situations. In this model creep is neglected, i.e. there is no friction between the two objects. If two objects have different elastic properties, then friction occurs which makes this assumption less realistic. Using the Hertz equations the displacements of the materials can be computed. Although the Hertz model does not describe reality perfectly, is still being used as of today.

It took a long time (until the 1960s) till research showed that the Hertz model wasn't completely accurate. This resulted in the development of new contact models. In 1970 the JKR model (named by its inventors Johnson, Kendall, and Roberts) was developed. This new model is an improvement of the Hertz model and 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 another improvement of the contact model and also includes the effect of plastic deformation.

The CONTACT software is capable of computing the pressure distributions and elastic displacements between two objects with general smooth geometries. It is also capable of taking rolling or sliding into account. The contact area and regions with adhesion or slip can be identified for many different problems.

Throughout the paper, we only consider the results of CONTACT and Hertz theory when applicable. In this chapter, we will focus on stationary contact problems, i.e. we will look at the equilibrium situation when two objects are being pressed upon each other using Hertz theory. For this equilibrium situation, the contact force and pressure distribution at the

contact area will be derived as function of the penetration. If friction is neglected, the same contact force and pressure distribution can be used for time-dependent contact problems.

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 it can be represented by $\{(x, y, z) : z \leq 0\}$ in the static situation without external forces). Now imagine a force F_z pushing downwards at the origin. The equilibrium solution satisfies the stationary elasticity equations with no external forces

$$\frac{\partial \sigma_{ij}}{\partial x_j} = 0 \qquad \text{for } i = 1, 2, 3 \tag{4.1}$$

The force F_z is described by a boundary condition for the traction at the top of the half space. Furthermore, since the half plane is infinity large, we have $\mathbf{u}(x, y, z) \to \mathbf{0}$ as $x^2 + y^2 + z^2 \to \infty$ for our other boundary conditions. Then, according to [1], the displacement caused by this force is:

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

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

$$u_z = \frac{1+\nu}{2\pi E}\left[\frac{2(1-\nu)}{r} - \frac{z^2}{r^3}\right] F_z \tag{4.4}$$

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

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

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

$$u_z = \frac{(1-\nu^2)}{\pi E r} F_z \tag{4.7}$$

Note that since $z = 0$ we also have $r^2 = x^2 + y^2$. Equation (4.5) and (4.6) 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 (4.7) 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 (4.7), the z-displacement at a point (x, y) on the surface is given by

$$u_z = \iint_A \frac{1-\nu^2}{\pi E} \cdot \frac{p}{r} \mathrm{d}u\mathrm{d}v = \frac{1}{\pi E^*} \iint_A \frac{p(u,v)}{\sqrt{(u-x)^2+(v-y)^2}} \mathrm{d}u\mathrm{d}v \tag{4.8}$$

where

$$E^* = \frac{E}{1-\nu^2} \tag{4.9}$$

A crucial part of Hertz theory tells how the pressure is distributed around the contact area. Let a be the radius of the contact area. Then p is assumed to be of the form

$$p(r) = \begin{cases} p_0 \left(1 - \dfrac{r^2}{a^2}\right)^{1/2} & \text{for } |r| \leq a \\ 0 & \text{for } |r| > a \end{cases} \tag{4.10}$$

The pressure is maximal at the center of the sphere and is 0 outside the contact area. The total force is by definition equal to

$$F = \int_0^a p(r) 2\pi r \mathrm{d}r = 2\pi p_0 \left[\frac{1}{3}a^2 \left(1 - \frac{r^2}{a^2}\right)^{3/2}\right]_0^a = \frac{2}{3}\pi p_0 a^2 \tag{4.11}$$

As derived in [4], if we substitute (4.10) into (4.8), we arrive at the displacement

$$u_z = \frac{\pi p_0}{4E^* a}(2a^2 - r^2) \tag{4.12}$$

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 surface is

$$z(x,y) = R - \sqrt{R^2 - x^2 - y^2} \approx \frac{x^2+y^2}{2R} \tag{4.13}$$

or similarly $z(r) \approx \frac{r^2}{2R}$. Let δ be the approach between the sphere and the surface. The approach is a scalar and is defined as the minimum vertical distance between the undeformed bodies. If the approach is positive, there is no contact yet between the two bodies. The approach is negative if and only if penetration between the two bodies occurs.

The vertical displacement at some point (x, y) is approximately

$$u_z = \delta - (R - \sqrt{R^2 - r^2}) \approx \delta - \frac{r^2}{2R} \tag{4.14}$$

where $r^2 = x^2 + y^2$. Equation (4.14) needs to be of the form (4.12). For this to be true, we must have

$$\delta = \frac{\pi p_0 a}{2E^*} \tag{4.15}$$

and furthermore

$$\frac{1}{2R} = \frac{\pi p_0}{4E^* a} \implies a = \frac{\pi p_0 R}{2E^*} \tag{4.16}$$

A direct consequence is that the contact radius satisfies

$$a^2 = R\delta \tag{4.17}$$

and we also can retrieve an explicit formula for p_0:

$$p_0 = \frac{2E^* a}{\pi R} = \frac{2E^*}{\pi R}\sqrt{R\delta} = \frac{2E^*}{\pi}\sqrt{\frac{\delta}{R}} \tag{4.18}$$

And finally by substituting equations (4.17) and (4.18) into (4.11) we arrive at the relation between the contact force F and the approach δ.

$$F = \frac{2}{3}\pi p_0 a^2 = \frac{2}{3}\pi\frac{2E^*}{\pi}\sqrt{\frac{\delta}{R}}Rd = \frac{4}{3}E^* R^{1/2}\delta^{3/2} \tag{4.19}$$

Similarly, by substituting (4.18) into (4.10) we arrive at the explicit pressure distribution

$$p(r) = \frac{2E^*}{\pi}\sqrt{\frac{\delta}{R}}\left(1 - \frac{r^2}{a^2}\right)^{1/2} \tag{4.20}$$

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 (see [4]) equations (4.17) - (4.19) remain true, as long as the radius R is chosen such that

$$\frac{1}{R} = \frac{1}{R_1} + \frac{1}{R_2} \tag{4.21}$$

4.4 Contact between elastic bodies

A simple adjustment can be made to apply the previous theory to the case with two curved 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

$$\frac{1}{E^*} = \frac{1-\nu_1^2}{E_1} + \frac{1-\nu_2^2}{E_2} \tag{4.22}$$

instead of equation (4.9), then the same formulas remain correct. Note that the approach δ here is distance between the two bodies if the deformation of both bodies is neglected.

4.5 Using CONTACT to compute p and F

According to Hertz theory, the pressure p and contact force F are described by equations (4.19) and (4.20). However, these equations are only valid for spherical objects.

CONTACT, however, is capable of working with bodies with more complex geometries. Using an input file, this geometry can be described which is then used by CONTACT for the internal computations. The user can either supply the penetration, or the so called undeformed distance as well as the contact force. The choice of the approach depends on the contact problem.

CONTACT can be used to solve many kinds of contact problems. It is able to identify the size and shape of the contact area and compute pressure distributions in this contact area as well as elastic displacements of the bodies. For the problems researched in this report, we generally assume that no friction occurs. CONTACT, however, can take friction into account and is able to compute tangential shear stresses. Furthermore, the regions with adhesion and with slip can be identified using CONTACT.

Additionally, CONTACT also computes the elastic displacement of the elements at the boundary. These displacements correspond to the stationary situation, i.e. they are the result of solving the stationary elasticity equations (4.1).

4.5.1 The penetration and undeformed distance

The penetration is defined as the distance between two undeformed objects. It can be measured at any point (x, y) and is denoted by $\delta(x, y)$. It can be written as

$$\delta(x, y) = z_1(x, y) - z_2(x, y) \tag{4.23}$$

where z_1 and z_2 are the height of the bottom and upper object, respectively.

In some situations, the rigid height of an object is not yet known. For these problems. we define the undeformed distance $h(x, y)$ as the geometric distance between the two bodies in its undeformed state. In this definition, however, it doesn't matter from which height the objects are measured. Unlike the penetration, the undeformed distance is hence defined up to a constant. Usually, this constant is chosen such that

$$0 = \min_{(x,y)\in\Omega} h(x, y) \tag{4.24}$$

If this definition is used, the difference between the penetration and the undeformed distance is equal to a constant, which is the approach.

$$\delta(x, y) = h(x, y) - \delta \tag{4.25}$$

The undeformed distance is useful in situations where the approach is not known. It represents the geometry of two objects which is in particular of importance for determining the contact area and the distribution of the pressure.

A 1-D example can be shown in Figure 4.1. This figure shows the shape of two arbitrary objects in their undeformed state at a certain point in time. Here, the approach is approximately $\delta \approx 0.2$mm. The red line represents the penetration. The undeformed distance can

also be represented by the red line. Even after translating this in the y-direction, the red line remains a valid representation for the undeformed distance.

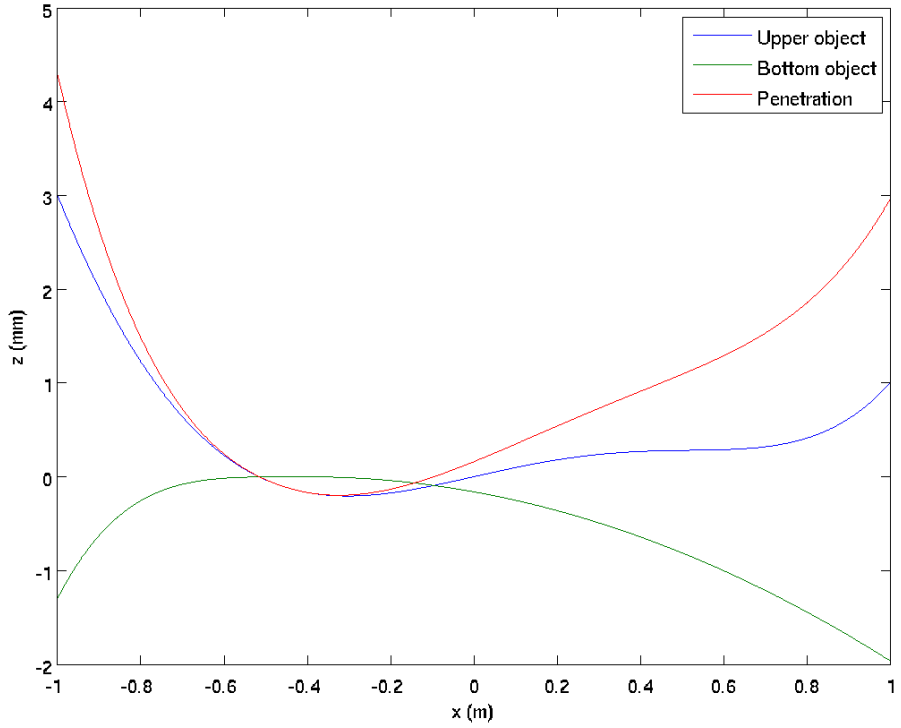


Figure 4.1: The penetration between two objects

4.5.2 Using CONTACT

CONTACT is capable of computing the pressure distribution in the contact area, the normal force as well as the elastic displacements at the surface near the contact area in two different ways. In both situations, the undeformed distance should be supplied to CONTACT. CONTACT can use this to determine the shape of the interacting bodies. Furthermore, either the normal force or the approach should be supplied. The two options are possible in different situations.

Consider the stationary problem of a sphere resting on an elastic half space. The contact force is simply equal to the force the gravity exerts on the sphere, i.e. $F_n = mg$, where m is the mass of the sphere.

In this situation, one is interested in computing the approach. This is not known beforehand. The contact force, however, is known, since we are looking at a stationary problem. If this contact force is supplied, then CONTACT is capable of computing the initial contact point, the pressure distribution around it as well as well as the static displacement of the surface elements near the contact area.

CONTACT requires the undeformed distance to be specified for a rectangular grid. This grid is divided by $m_x \times m_y$ elements. The horizontal distance between the elements is Δx and the vertical distance is Δy. These variables can be chosen by the user. A smaller Δx or Δy will result in an improved accuracy, but this is more resource expensive for CONTACT. It is important that the user makes sure that this rectangular grid contains the whole contact area. CONTACT returns the pressure and static displacements in the same grid points as the user supplies.

This approach is useful for stationary problems where the contact force is known beforehand. This is not the case for time-dependent problems, since the normal force does not have to be equal to the gravitational force. For time-dependent problems, however, the approach is known at each time step. If the approach is supplied, then CONTACT is capable of determining the contact force, the pressure as well as the static displacements of the boundary elements.

For these kind of problems, the approach and the undeformed distance, and thus the penetration, are both known at each time step. The contact force F_n, however, is not. CONTACT can therefore be used to compute the normal force F_n and the pressure as a function of the approach δ.

4.5.3 Example input file

To run CONTACT, an input file needs to be created. This input file contains information such as the elasticity parameters, size and mesh width of the grid, lots of CONTACT parameters representing the type of contact problem, as well as the undeformed distance combined with the contact force or the penetration. After running CONTACT using this input file as parameter, several output files are created. Most importantly, these output file contain the contact force, as well as the elastic translation and pressure at each grid point.

As an example, suppose that we are interested in the usual Hertz problem. Consider a rigid sphere of radius $R = 0.10$m. The sphere penetrates an elastic half space with Young's modulus $E = 1.5 \cdot 10^8$, and Poisson's ratio $\nu = 0.5$ for a total of 5mm. The following input file can be used.

```
 3 module % result element 1, Contact patch 1
% Next case 1
  200020 P−B−T−N−F−S PVTIME, BOUND , TANG , NORM , FORCE , STRESS
  022020 L−D−C−M−Z−E FRCLAW, DISCNS, INFLCF, MATER , RZNORM, EXRHS
  002111 G−I−A−O−W−R GAUSEI, IESTIM, MATFIL, OUTPUT, FLOW , RETURN
   200 30 30 1 1.0E−04 MAXGS , MAXIN , MAXNR , MAXOUT, EPS
  0.000 0.000 0.000 0.000 FUN, FUX, FUY, CPHI
% Note: N=1 means FUN == FN, F=0 means FUX == CKSI, FUY == CETA
  0.3000 0.3000 FSTAT, FKIN
  0.5000 0.5000 5.0000E+07 1.0000E+20 SIGMA 1,2, GG 1,2
    1 IPOTCN
   11 11 −0.028 −0.03 0.0050 0.0050 MX,MY,XL,YL,DX,DY
    9 1 IBASE, IPLAN
% PENETRATION, (1)−(2): SPECIFIED PER ELEMENT

    1.4586E−03 2.6352E−04 −6.5563E−04 −1.3068E−03 −1.6954E−03
    −1.8246E−03 −1.6954E−03 −1.3068E−03 −6.5563E−04 2.6352E−04
    1.4586E−03
```

```
19
20      2.6352E-04 -9.1663E-04 -1.8246E-03 -2.4679E-03 -2.8519E-03
21      -2.9796E-03 -2.8519E-03 -2.4679E-03 -1.8246E-03 -9.1663E-04
22      2.6352E-04
23
24      -6.5563E-04 -1.8246E-03 -2.7241E-03 -3.3616E-03 -3.7421E-03
25      -3.8686E-03 -3.7421E-03 -3.3616E-03 -2.7241E-03 -1.8246E-03
26      -6.5563E-04
27
28      -1.3068E-03 -2.4679E-03 -3.3616E-03 -3.9949E-03 -4.3730E-03
29      -4.4987E-03 -4.3730E-03 -3.9949E-03 -3.3616E-03 -2.4679E-03
30      -1.3068E-03
31
32      -1.6954E-03 -2.8519E-03 -3.7421E-03 -4.3730E-03 -4.7497E-03
33      -4.8749E-03 -4.7497E-03 -4.3730E-03 -3.7421E-03 -2.8519E-03
34      -1.6954E-03
35
36      -1.8246E-03 -2.9796E-03 -3.8686E-03 -4.4987E-03 -4.8749E-03
37      -5.0000E-03 -4.8749E-03 -4.4987E-03 -3.8686E-03 -2.9796E-03
38      -1.8246E-03
39
40      -1.6954E-03 -2.8519E-03 -3.7421E-03 -4.3730E-03 -4.7497E-03
41      -4.8749E-03 -4.7497E-03 -4.3730E-03 -3.7421E-03 -2.8519E-03
42      -1.6954E-03
43
44      -1.3068E-03 -2.4679E-03 -3.3616E-03 -3.9949E-03 -4.3730E-03
45      -4.4987E-03 -4.3730E-03 -3.9949E-03 -3.3616E-03 -2.4679E-03
46      -1.3068E-03
47
48      -6.5563E-04 -1.8246E-03 -2.7241E-03 -3.3616E-03 -3.7421E-03
49      -3.8686E-03 -3.7421E-03 -3.3616E-03 -2.7241E-03 -1.8246E-03
50      -6.5563E-04
51
52      2.6352E-04 -9.1663E-04 -1.8246E-03 -2.4679E-03 -2.8519E-03
53      -2.9796E-03 -2.8519E-03 -2.4679E-03 -1.8246E-03 -9.1663E-04
54      2.6352E-04
55
56      1.4586E-03 2.6352E-04 -6.5563E-04 -1.3068E-03 -1.6954E-03
57      -1.8246E-03 -1.6954E-03 -1.3068E-03 -6.5563E-04 2.6352E-04
58      1.4586E-03
59  % UNRESTRICTED PLANFORM
60    0 module
```

For more information about all the input parameters, one could consult the CONTACT user guide [5]. In our case, we used $N = 0$. This parameter specifies whether the normal force or the approach is prescribed. We used $N = 0$ and prescribed the approach $\delta = 0.005$m.

It is worth noting that CONTACT requires both objects to be non-rigid. To circumvent this, we model the rigid sphere as an elastic sphere with a very large Young's modulus $E > 10^{20}$. In this way, the sphere behaves like a rigid object.

4.6 Comparison between Hertz theory and CONTACT

Given the approach for the Hertz problem, the contact force can be computed using Hertz theory and is given by equation (6.1). The pressure distribution around the contact area is given by (4.10). Alternatively, the same can be computed by using CONTACT. Given the maximum approach δ measured from the lowest point of the sphere, the penetration between the surface and the sphere at any given point (x, y) is

$$\delta(x, y) = \sqrt{R^2 - x^2 - y^2} - \delta \qquad \text{for } x^2 + y^2 \leq R^2 \tag{4.26}$$

This penetration can then be supplied to CONTACT, similarly to the example input file as given before. The contact force can then be compared with the result of the Hertz problem, see Figure 4.2. Clearly, the contact force computed using CONTACT corresponds to the Hertz solution. As the penetration increases, however, the results become slightly off. This is because the penetration supplied to CONTACT was computed for a mesh. CONTACT interpolates this penetration for internal points. As the mesh becomes finer, this interpolation becomes better, and the result becomes more accurate.

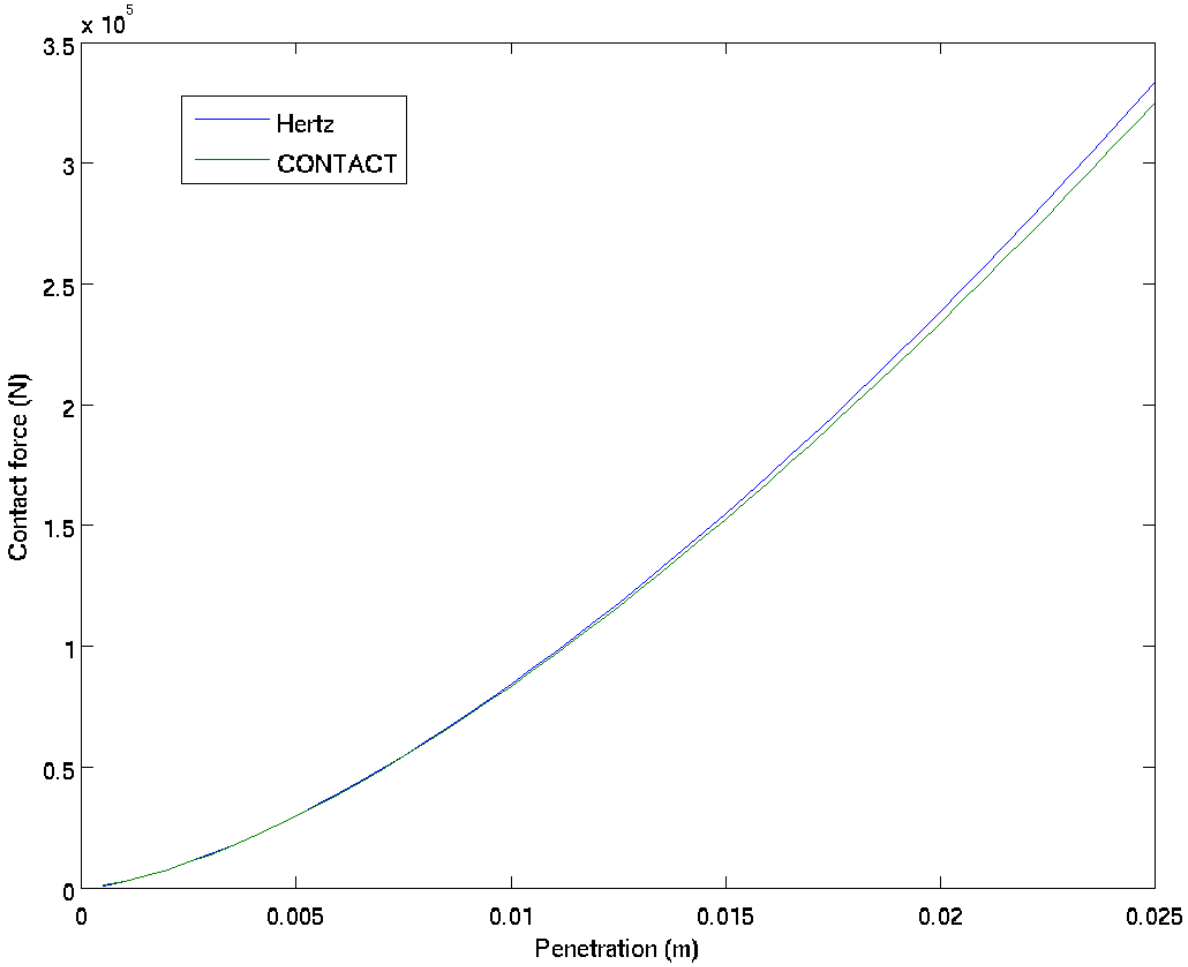


Figure 4.2: Comparison of the contact force computed using Hertz theory and CONTACT.

Chapter 5

Numerical methods for dynamical contact problems

5.1 Newton's equations of motion for contact problems

For contact dynamics, Newton's equations of motion play an important role. Consider the following example (which will be more thoroughly discussed in Chapter 6). Consider a small object with mass m that penetrates an elastic surface. The height of the object can be represented by $z(t)$. One is interested in determining this height.

If the object penetrates the surface, the surface will indent slightly. A contact force F_n upwards will then be exerted by the surface as discussed in Chapter 4. This force, however, becomes larger as more penetration occurs and is hence a function of the height of the object. The resulting force therefore is $F(z) = F_n(z) - F_g$. By Newton's second law of motion, this resulting force is proportional to the acceleration of the object, i.e.

$$F(z) = m\frac{\mathrm{d}^2 z}{\mathrm{d}t^2} \tag{5.1}$$

Newton's equations of motion are crucial for dynamic contact problems. In general, the corresponding equations of motion for linear structural dynamic problems in multiple dimensions have the form

$$M\ddot{\mathbf{x}} + C\dot{\mathbf{x}} + K\mathbf{x} = \mathbf{F}(t) \tag{5.2}$$

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 have discussed in Chapter 4. For spherical contact on an elastic-half plane, we have seen that contact force F_n is proportional to $\delta^{3/2}$ as long as $\delta \geq 0$.

The most general form of the Newton's equation of motion is

$$M(\mathbf{x})\ddot{\mathbf{x}} + P(\mathbf{x}, \dot{\mathbf{x}}) = \mathbf{F}(t) \tag{5.3}$$

with M and P being any function.

5.2 Forward / Backward Euler

To solve differential equation (5.3), integration schemes are required. For simplicity and completeness, we will start with the well known Forward and Backward Euler integration schemes. These schemes allow to integrate first order ordinary differential equations with respect to time. To apply this, we first transform equation (5.3) to a (twice as large) system of first order differential equations. We introduce $\mathbf{y} = \dot{\mathbf{x}}$ so that

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{y} \\ \dot{\mathbf{y}} &= M^{-1}(\mathbf{x})(\mathbf{F}(t) - P(\mathbf{x}, \mathbf{y})) \end{aligned} \tag{5.4}$$

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

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

$$\mathbf{z}_{k+1} = \mathbf{z}_k + \Delta t \mathbf{g}(t_k, \mathbf{z}_k) \tag{5.5}$$

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

$$\mathbf{z}_{k+1} = \mathbf{z}_k + \Delta t \mathbf{g}(t_{k+1}, \mathbf{z}_{k+1}) \tag{5.6}$$

Often, because $\mathbf{g}$ can be a complex non-linear 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 $\mathbf{z}^0_{k+1} = \mathbf{z}_k$ as a good initial guess and repeatedly compute

$$\mathbf{z}^{j+1}_{k+1} = \mathbf{z}_k + \Delta t \mathbf{g}(t_{k+1}, \mathbf{z}^j_{k+1}) \tag{5.7}$$

iteratively until $\|\mathbf{z}^{j+1}_{k+1} - \mathbf{z}^j_{k+1}\| < \varepsilon$ for a certain error margin. Alternatively, one can apply a faster converging iterative method like Newton-Raphson or the secant method.

5.3 Runge Kutta / Radau methods

The Runge-Kutta methods are generalizations of the Forward/Backward Euler 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

$$\mathbf{z}_{k+1} = \mathbf{z}_k + \Delta t \sum_{i=1}^{n} b_i \mathbf{k}_i \tag{5.8}$$

where

$$\mathbf{k}_i = \mathbf{g}(t_k + c_i\Delta t, \mathbf{z}_k + \Delta t \sum_{j=1}^{n} a_{ij}\mathbf{k}_j) \tag{5.9}$$

The components of the matrix A (with components a_{ij}) and vectors $\mathbf{c}$ and $\mathbf{b}$ are often compactly written by using a Butcher tableau. Each Runge-Kutta/Radau scheme can be described by one.

The Butcher tableau corresponding to Runge-Kutta methods are strictly lower triangular matrix. As a result, computing $\mathbf{k}_i$ does not require the use of $\mathbf{k}_j$ for $j \geq i$. Hence the method is explicit. As an example, for the widely used RK4 method, for example, we have the tableau

$$\begin{array}{c|cccc} c_1 & a_{11} & a_{12} & \dots & a_{1n} \\ c_2 & a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \vdots & & \vdots \\ c_b & a_{n1} & a_{n2} & \dots & a_{nn} \\ \hline & b_1 & b_2 & \dots & b_n \end{array} = \begin{array}{c|cccc} 0 & 0 & 0 & 0 & 0 \\ 1/2 & 1/2 & 0 & 0 & 0 \\ 1/2 & 0 & 1/2 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ \hline & 1/6 & 1/3 & 1/3 & 1/6 \end{array} \tag{5.10}$$

Using (5.8) and (5.9), 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 Runge-Kutta methods can be used to achieve a high order accuracy at the cost of more function evaluations. Using a Runge-Kutta method of n stages (i.e. the Butcher tableau has size $n \times n$), an accuracy of $\mathcal{O}(\Delta t^n)$ can be achieved. The disadvantage of these schemes is that the corresponding region of stability is small; this is especially a problem for stiff differential equations which we will often consider.

The matrix A, representing the Butcher tableau of the integration scheme, does not have to be a strictly lower triangular matrix. If this is not the case, then the corresponding integration scheme is implicit. The Radau methods are fully implicit Runge-Kutta schemes and can be seen as generalisations of Backward Euler. As described in [6], the default Radau5 method is given by the Butcher tableau

$$\begin{array}{c|ccc} 0 & \frac{1}{9} & \frac{-1-\sqrt{6}}{18} & \frac{-1+\sqrt{6}}{18} \\ \frac{3}{5}-\frac{\sqrt{6}}{10} & \frac{1}{9} & \frac{11}{45}+\frac{7\sqrt{6}}{360} & \frac{11}{45}-\frac{43\sqrt{6}}{360} \\ \frac{3}{5}+\frac{\sqrt{6}}{10} & \frac{1}{9} & \frac{11}{45}+\frac{43\sqrt{6}}{360} & \frac{11}{45}-\frac{7\sqrt{6}}{360} \\ \hline & \frac{1}{9} & \frac{4}{9}+\frac{\sqrt{6}}{36} & \frac{4}{9}-\frac{\sqrt{6}}{36} \end{array} \tag{5.11}$$

Since the Radau methods are implicit, an iterative scheme should be used to determine the solutions $\mathbf{z}^{k+1}$. Additionally, each iteration requires the evaluation of multiple function evaluations, which can therefore be very costly. The main advantage is that all Radau methods are A-stable, and can hence be used for stiff problems. Additionally, the accuracy of the Radau methods is $\mathcal{O}(\Delta t^{2n-1})$, where n is the number of stages.

5.4 The Verlet method

The Verlet method [7] is an explicit integration scheme can be used to integrate the second order differential equation of the form $\ddot{\mathbf{x}} = \mathbf{g}(\mathbf{x})$. The method is of order $\mathcal{O}(\Delta t)$ accurate and is given by

$$\begin{aligned} \mathbf{x}_1 &= \mathbf{x}_0 + \mathbf{v}_0 \Delta t + \frac{(\Delta t)^2}{2} \mathbf{g}(\mathbf{x}_0), \text{ and} \\ \mathbf{x}_{k+1} &= 2\mathbf{x}_k - \mathbf{x}_{k-1} + (\Delta t)^2 \mathbf{g}(\mathbf{x}_k) \qquad \text{for } k \geq 1 \end{aligned} \tag{5.12}$$

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 it is much more energy conserving compared with other explicit methods with a similar order of convergence such as Forward Euler. Although the method is explicit and we do not keep track of the velocity components, the Verlet scheme keeps track of the total amount of energy. This is explained in [8] and is in particular due to the symplecticity and time-reversibility of the Verlet scheme. We will also show this behaviour numerically in Chapter 6. The method, however, it is not always applicable for the most general case (5.3), for example when there is air resistance, so that $\mathbf{g}$ is also a function of $\dot{\mathbf{x}}$.

5.5 Leapfrog integration

Leapfrog integration is similar to the Verlet method. It is an explicit scheme usable for differential equations of the form $\ddot{\mathbf{x}} = \mathbf{g}(\mathbf{x})$. Similarly to the Verlet method, Leapfrog integration is very energy conserving since the method is symplectic and time-reversible. It is, however, of $\mathcal{O}(\Delta t^2)$ accurate, yet the number of function evaluations per time step remains the same as for Verlet or Forward Euler.

The leapfrog method keeps track of both the displacement and velocity components. A unique property of the Leapfrog method is that the displacement and velocity components 'leapfrog' over each other. That is, the velocity components are described exactly half way between two consecutive displacement components. It is defined by

$$\begin{aligned} \mathbf{x}_{k+1} &= \mathbf{x}_k + \Delta t \mathbf{v}_{k+1/2} \\ \mathbf{v}_{k+3/2} &= \mathbf{v}_{k+1/2} + \Delta t \mathbf{g}(\mathbf{x}_{k+1}) \end{aligned} \tag{5.13}$$

for $k \geq 1$. Here, $\mathbf{v}$ represent the velocity components. Note that (5.13) is not self-starting, since the component $\mathbf{v}_{1/2}$ is not known. This term can, however, be approximated using a self-starting scheme such as Forward Euler.

A disadvantage of the Leapfrog method is that the time step Δt must remain constant. This is required [9] to maintain stability.

5.6 Newmark's method

The advantage of the Verlet and Leapfrog method is that the methods are energy conserving. However, both schemes are explicit. This is a problem since explicit methods are unstable for very stiff problems.

Newmark's method [10] (also known as the Newmark-beta method) is a popular integration scheme for problems in structural dynamics. The method is useful because it not only does it preserve energy, it is also implicit; hence the method has better stability properties than Verlet or Leapfrog.

$$\begin{aligned} \mathbf{x}_{k+1} &= \mathbf{x}_k + \Delta t \mathbf{v}_k + \frac{(\Delta t)^2}{2}((1-2\beta)\mathbf{a}_k + 2\beta \mathbf{a}_{k+1}) \\ \mathbf{v}_{k+1} &= \mathbf{v}_k + \Delta t((1-\gamma)\mathbf{a}_k + \gamma \mathbf{a}_{k+1}) \end{aligned} \tag{5.14}$$

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 (5.3) by

$$M(\mathbf{x}_{k+1})\mathbf{a}_{k+1} + P(\mathbf{x}_{k+1}, \mathbf{v}_{k+1}) = \mathbf{F}(t_{k+1}) \tag{5.15}$$

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 $[t_k, t_{k+1}]$.

5.7 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 [11] is an extension of the Newmark method. An extra parameter α is introduced. The equations (5.14) remain the same, however, instead of the discrete Newton's equation of motion (5.15) the HHT method is slightly different:

$$M(\mathbf{x}_{k+1})\mathbf{a}_{k+1} + (1-\alpha)P(\mathbf{x}_{k+1}, \mathbf{v}_{k+1}) + \alpha P(\mathbf{x}_k, \mathbf{v}_k) = F((1-\alpha)t_{k+1} + \alpha t_k) \tag{5.16}$$

If $\alpha = 0$, then the term $\alpha P(\mathbf{x}_k, \mathbf{v}_k)$ drops out of (5.17) 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.

5.8 The generalized-α method

The generalized-α method [12] is again a generalisation of the HHT method. Once again the equations (5.14) 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:

$$\begin{aligned}&(1-\alpha_M)M(\mathbf{x}_{k+1})a_{k+1} + \alpha_M M(\mathbf{x}_k)a_k + \\ &(1-\alpha_F)P(\mathbf{x}_{k+1},\mathbf{v}_{k+1}) + \alpha_F P(\mathbf{x}_k,\mathbf{v}_k) = F((1-\alpha_F)t_{k+1} + \alpha_F t_k)\end{aligned} \tag{5.17}$$

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 $\beta = (1 - \alpha_M + \alpha_F)^2/4$.

5.9 Adams methods

Most schemes in the previous sections are similar in the fact that each is a single-step scheme; that is, to compute a component at time step $k+1$ we only need to know the components at the previous time step k. Multi-step schemes make use of earlier components that have already been computed at iteration $k-1$ and before. This information can be used to have a better approximation of the acceleration. This is especially useful for stiff differential equations where energy conservation is of importance.

Adams methods [13] are multistep integration schemes. They are capable of solving the differential equation $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, t)$; hence a transformation as in Section 5.2 should be made to transform to a system of first order differential equations. The solution at the next time step is approximated by using a polynomial of order n.

A general multi-step scheme of n steps has the form

$$\sum_{i=0}^{n} a_i \mathbf{x}_{k+i} = \Delta t \sum_{i=0}^{n} b_i \mathbf{f}(\mathbf{x}_{k+i}, t_{k+i}) \tag{5.18}$$

Usually, we assume that $a_n = 1$ by simply scaling this equation. Hence, after time step $k+n-1$ we can set

$$\mathbf{x}_{k+n} = \Delta t \sum_{i=0}^{n} b_i \mathbf{f}(\mathbf{x}_{k+i}, t_{k+i}) - \sum_{i=0}^{n-1} a_i \mathbf{x}_{k+i} \tag{5.19}$$

There are different families of Adams methods. The Adams-Bashforth methods are explicit methods; they require that $a_{n-1} = -1$ and $a_i = 0$ for $i \neq n-1$. The coefficients b_i define the integration scheme. For Adams-Bashforth methods it is required that $b_n = 0$ since the methods are explicit. The Adams-Moulton methods are implicit Adams method and do not have a restriction on b_n.

5.10 Backward differentiation formulas

The Backward differentiation formulas [14] are another family of implicit multistep integration schemes. They can also be described as in (5.19). Compared with the Adams-Moulton methods, however, the coefficients b_i for $i = 0$ to $n-1$ are required to be zero. Hence, the expression contains only a single function evaluation. On the other hand, the coefficients a_i for $i = 0$ to $n-1$ are not required to be zero; these are chosen to achieve the highest order of accuracy.

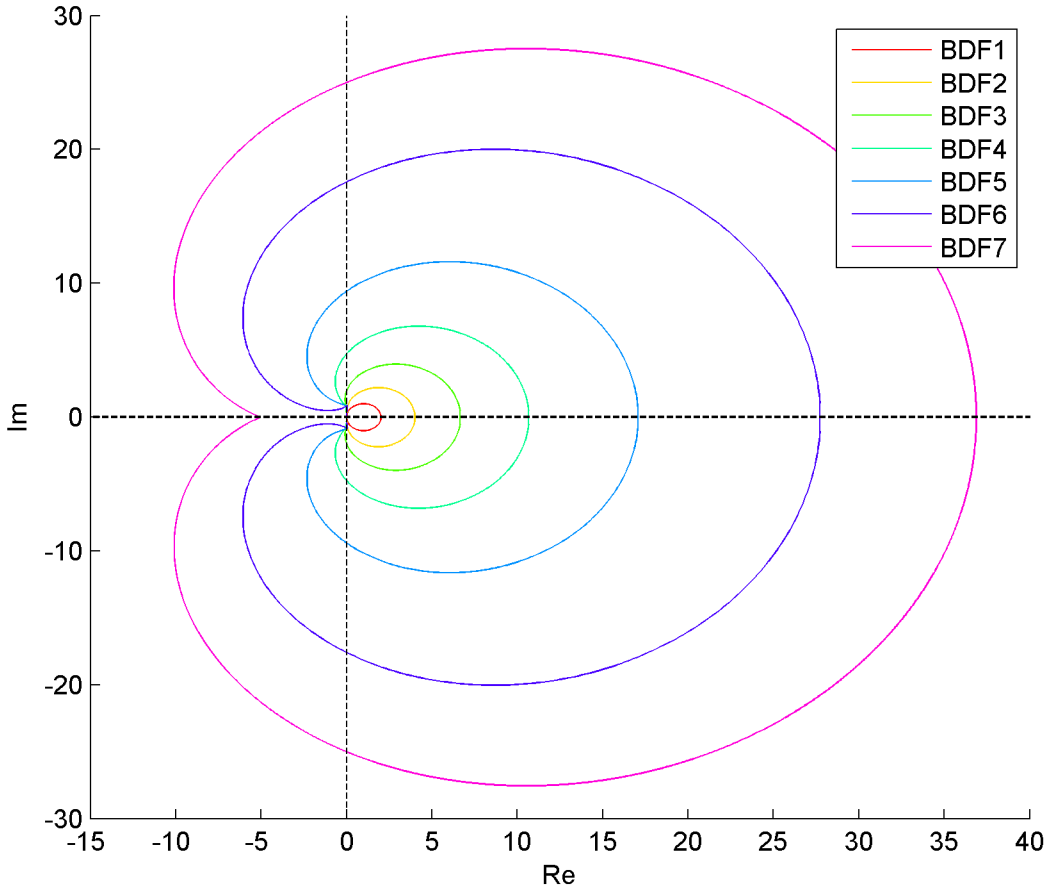


Figure 5.1: The stability region for different backward differentiation formulas. Note that stability is achieved outside these circles, similarly to the stability of the Backward Euler (BDF1) method.

As an example, for $n = 1$ we have the usual Backward Euler scheme. For $n = 2$, we have the components $a_0 = \frac{1}{3}, a_1 = -\frac{4}{3}$ and $\beta = \frac{2}{3}$.

When using Backward differentiation formulas, a very important thing to realise is the stability of the method. See Figure 5.1 for the stability region of the first 6 Backward difference formulas. The first two methods are A-stable, but as the order of the Backward differen-

tial formula increases, the method becomes more and more unstable. The order of the used Backward differentiation formulas should therefore be chosen carefully in order to preserve stability while keeping the numerical scheme accurate. Backward difference formula of order $\mathcal{O}(\Delta t^n)$ for $n \geq 7$ are of no use, since the contour of the stability region crosses the negative real axis.

Chapter 6

Rigid body motion

The goal of this chapter is to get a feeling of how dynamical contact problems can be derived and solved using a time integration scheme such as the ones we have discussed in Chapter 5. We will consider a simple contact problem involving a sphere falling on an elastic surface, as we have already shortly mentioned in the beginning of Chapter 5. This problem will result in a one-dimensional differential equation for the height of the sphere. Solving this differential equation requires the use of a time-integration scheme. Multiple time integration schemes as described in Chapter 5 will be used to solve the problem. This will allow us to determine basic properties of the numerical schemes that occur for dynamical contact problems such stability and energy conservation.

Consider a solid sphere of radius R which is dropped from height z_0 above an elastic half-space having Young's modulus E and Poisson's ratio ν. Before dropping this sphere, it has no vertical speed, i.e. $v_0 = 0$. The mass of the ball is simply $m = \frac{4}{3}\pi R^3 \rho$, where ρ is the density of the material of the sphere.

The sphere is supposed to be either rigid or elastic, but in the latter case we assume that the modulus of elasticity of the ball is the same as for the half-plane. This is required, since otherwise friction would occur during contact. 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.

For now, we are only interested in the rigid height of the sphere as a function of the time. The height is measured from the lowest point of the sphere and is allowed to be negative. The height $z(t)$ is therefore equal to $z(t) = -\delta(t)$, with $\delta(t)$ being the approach.

Since Hertz theory describes the normal force between two spheres or between a sphere and an elastic half space, the rigid height of the sphere can easily be solved. If one is interested in the deformation of the surface as function of the time, one could apply the quasi-static approach as will be described in Section 6.4. This approach, however, is not very accurate since the inertia of particles at the boundary elements of the half-space is ignored. In Chapter 7 we will describe a different method to solve this problem. This method takes inertia into account but is computationally expensive.

6.1 Derivation of the differential equation

Equation (4.19) gives a relation between the approach 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.

At all times, gravity exerts a force on the sphere pointing downwards. This force is constant and is given by $F_g = -mg$. The minus sign is important since the positive z direction points upwards. 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 $z(t)$ (which is negative if and only if there is penetration), then the approach is $\delta(t) = -z(t)$. Equation (4.19) is valid only if the approach is positive; if the approach would be negative, there would be no contact between the sphere and the half-plane and therefore the half-plane would not exert any force upwards. Equation (4.19) can hence be formulated as

$$F_n = \frac{4}{3} E^* R^{1/2} \max(0, -z)^{3/2} \tag{6.1}$$

So that the resulting force is

$$F(z) = F_n + F_g = \frac{4}{3} E^* R^{1/2} \max(0, -z)^{3/2} - mg \tag{6.2}$$

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

$$\ddot{z} = \frac{4}{3m} E^* R^{1/2} \max(0, -z)^{3/2} - g \tag{6.3}$$

This is a non-linear ordinary differential equation of form (5.3), where $M \equiv 1$, $F \equiv -g$ and $P(z) = -\frac{4}{3m} E^* R^{1/2} \max(0, -z)^{3/2}$.

6.2 Properties of the solution

It is very hard (or even impossible) to solve differential equation (6.3) analytically. However, it is easy to derive some basic properties of the solution. Let t_0 be the first moment the ball touches the surface. Before this point, equation (6.3) can be simplified to

$$\ddot{z} = -g \tag{6.4}$$

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

$$z(t) = z_0 - \frac{1}{2} g t^2 \tag{6.5}$$

for $0 \leq t \leq t_0$. Equation (6.5) describes the height of the sphere during its free fall. It follows that $t_0 = \sqrt{2 z_0}$. However, after touching the surface, the differential equation becomes non-linear and very hard to solve analytically. By substituting $w := -z$ and defining $a := -\frac{4 E^* R^{1/2}}{3m}$ we can retrieve the following by using separation of variables:

$$\begin{aligned}
&\frac{\mathrm{d}^2 w}{\mathrm{d}t^2} = aw^{3/2} + g \\
\implies & w^{-3/2}\frac{\mathrm{d}^2 w}{\mathrm{d}t^2} = a + gw^{-3/2} \\
\implies & w^{-3/2} d^2 w = (a + gw^{-3/2}) dt^2 \\
\implies & \frac{w^{-3/2}}{a + gw^{-3/2}} d^2 w = dt^2
\end{aligned} \tag{6.6}$$

We can now try to integrate equation (6.6) 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, calculating the inverse of this function is extremely hard if not impossible. Therefore, we won't be able to express w in an explicit way as function of t.

We do know, however, that at some point $t_1 > t_0$, the ball reaches the lowest point. At this moment, the kinetic energy of the ball is zero. Since there is no damping in the system and therefore no energy is lost, 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.

6.3 Solving the problem numerically

Next, we will discuss the numerical methods as described in Chapter 5 applied for this test problem. All of the methods have been applied for this problem, but we only discuss some findings since many of the results are similar.

First of all, it is important to realise that differential equation (6.3) is non-linear. This has no negative impact on the use of explicit time integration schemes. For implicit time integration schemes such as Backward Euler, however, it usually becomes impossible to express the height z_{k+1} at the next iteration explicitly. If the differential equation would have been linear in z, we could have moved this term to the left side so that the only task at each iteration is to solve a 2×2 linear system.

Therefore, we have implemented a Picard iteration in order to solve the implicit expression. For example, Backward Euler integration for our problem is defined by

$$\begin{aligned}
z^{k+1} &= z^k + \Delta t v^{k+1} \\
v^{k+1} &= v^k + \Delta t \left[\frac{4}{3m} E^* R^{1/2} \max(0, -z^{k+1})^{3/2} - g \right]
\end{aligned} \tag{6.7}$$

The linear term $\Delta t v^{k+1}$ can be moved to the left side to get (in matrix-vector notation)

$$\begin{pmatrix} 1 & -\Delta t \\ 0 & 1 \end{pmatrix} \begin{pmatrix} z \\ v \end{pmatrix}^{k+1} = \begin{pmatrix} z \\ v \end{pmatrix}^{k} + \Delta t \begin{pmatrix} 0 \\ \frac{4}{3m} E^* R^{1/2} \max(0, -z^{k+1})^{3/2} - g \end{pmatrix} \tag{6.8}$$

Picard iteration can therefore be applied in the following way. We set

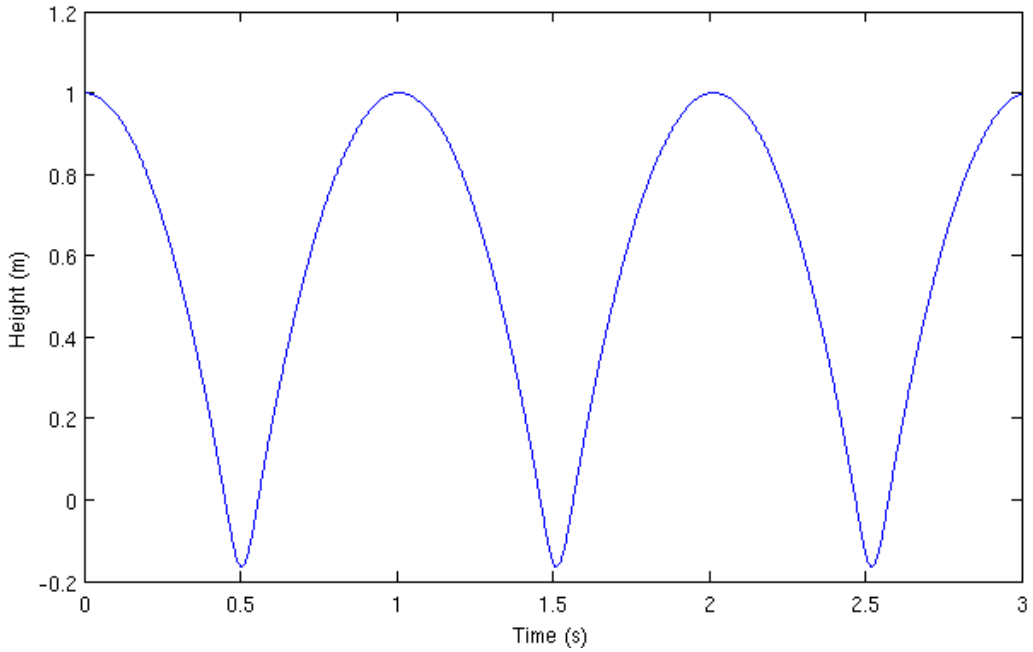


Figure 6.1: The height of the ball as a function of the time, using a stable method such as Radau5.

$$\begin{pmatrix} z \\ v \end{pmatrix}_{j+1}^{k+1} = \begin{pmatrix} 1 & -\Delta t \\ 0 & 1 \end{pmatrix}^{-1} \left[\begin{pmatrix} z \\ v \end{pmatrix}^{k} + \Delta t \begin{pmatrix} 0 \\ \frac{4}{3m} E^* R^{1/2} \max(0, -z_j^{k+1})^{3/2} - g \end{pmatrix} \right] \tag{6.9}$$

iteratively for each j until convergence is reached; e.g. when $\|[z;v]_{j+1}^{k+1} - [z;v]_j^{k+1}\| < \varepsilon$ for some small $\varepsilon > 0$. In our test problem, this iterative process always seemed to converge as long as Δt remains reasonable (not larger than 1, for example).

The Radau5 method yields the result as we would expect, as can be seen in Figure 6.1. 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 and 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. 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 see in Figure 6.2. The cyan line represents the total amount of energy, which is the sum of the kinetic, gravitational, and elastic energy. As one can see, the total amount of energy does indeed remain constant.

Energy conservation is an important property that is often important to maintain. As we have seen, this can be achieved by applying Radau5 for a small time step. Radau5, however, is a relatively expensive method compared with other integration schemes. We will therefore have a closer look and compare the energy conservation properties of different schemes.

To do so, we will solve (6.3) using multiple integration schemes. In each of the experiments, we consider a solid ball of radius $R = 0.1m$ and a mass of 10kg that is being dropped from $1m$.

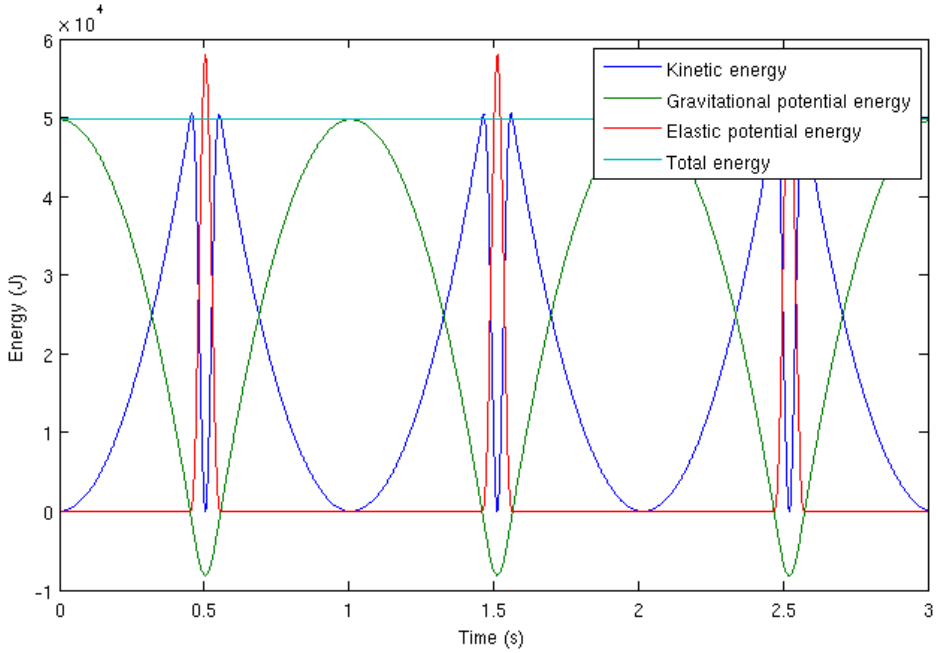


Figure 6.2: The energy of the ball as a function of the time, using a stable method such as Radau5.

For each method, a constant time step $\Delta t = 0.001$ is used. We vary the coefficient E. Then, the relative difference of the total amount of energy in the system compared with the total energy at the beginning is computed for different integration schemes. If this factor is larger than one, it means that amplification occurs for this integration scheme. A value smaller than one corresponds to to energy dissipation. Table 6.1 shows the maximum amplification (or dissipation) factor during the first 10 seconds of the integration. Note that in this table, we have subtracted one from the amplification factor.

Integrator	Accuracy	E		
		$2 \cdot 10^7$	$2 \cdot 10^9$	$2 \cdot 10^{11}$
Forward Euler	$\mathcal{O}(\Delta t)$	$3,925$	$71,96$	$268,7$
Backward Euler	$\mathcal{O}(\Delta t)$	$-0,764$	$-0,9998$	$-0,9999$
Verlet	$\mathcal{O}(\Delta t)$	$1,091 \cdot 10^{-3}$	$9,432 \cdot 10^{-4}$	$-0,186$
Leapfrog	$\mathcal{O}(\Delta t^2)$	$2,127 \cdot 10^{-5}$	$6,197 \cdot 10^{-4}$	$-0,207$
Newmark	$\mathcal{O}(\Delta t^2)$	$-1,733 \cdot 10^{-5}$	$-2,011 \cdot 10^{-3}$	$-0,0332$
BDF3	$\mathcal{O}(\Delta t^3)$	$-5,557 \cdot 10^{-4}$	$0,1186$	$2,401$
Radau3	$\mathcal{O}(\Delta t^3)$	$-2,676 \cdot 10^{-5}$	$5,603 \cdot 10^{-3}$	$-0,745$
RK4	$\mathcal{O}(\Delta t^4)$	$8,160 \cdot 10^{-4}$	$0,212$	1010
Radau5	$\mathcal{O}(\Delta t^5)$	$1,581 \cdot 10^{-6}$	$4,333 \cdot 10^{-4}$	$-0,116$

Table 6.1: The maximum energy amplification / dissipation factor after 10 seconds for different integration schemes.

Clearly, the explicit Runge-Kutta methods are unstable and are of no use for stiff problems. The result of Forward Euler can be seen in Figure 6.3. 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. Interestingly, the Verlet and Leapfrog scheme are also explicit methods but have a lot of energy conservation. However, if $E = 2 \cdot 10^{11}$ (i.e. we consider a steel half-space), then the disadvantage of the explicitness of both methods becomes visible.

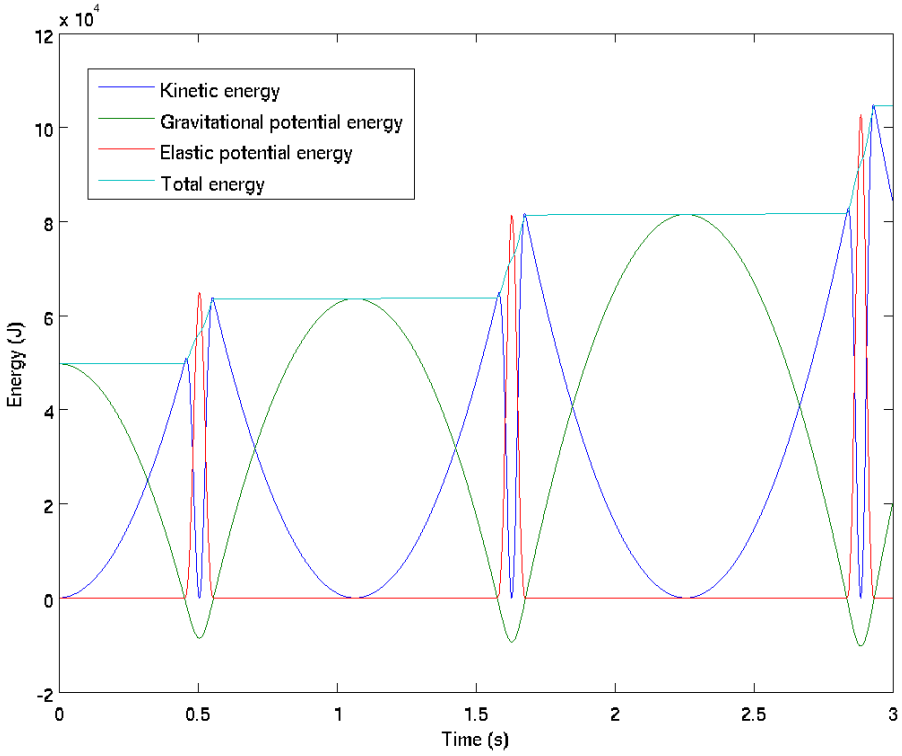


Figure 6.3: The energy of the ball as a function of the time, using the instable Forward Euler method.

For implicit methods such as the Radau methods, numerical damping will generally occur. This effect, however, is small if a method of high order is combined with a small time step. However, this is computationally expensive because many function evaluations will be needed.

The overall best integrator seems to be the Newmark-beta method. It is energy conserving similarly to the Verlet and Leapfrog method, but is also very stable, even for very stiff differential equations; this is because unlike the Verlet or Leapfrog method, the Newmark-beta method is implicit. Even if the half-space is made out of steel, only a small amount of numerical damping (or amplification) is created for the Newmark-beta method.

6.4 Using CONTACT to solve the problem

It is interesting to see if this problem can also be solved by using CONTACT. Instead of using equation (4.19) 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 the 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

$$Z_{ij} = R - \sqrt{R^2 - (X_{ij}^2 + Y_{ij}^2)} + z \tag{6.10}$$

As before, z denotes the rigid height of the ball, measured from its lowest point at $x = y = 0$. Note that equation (6.10) 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. It is very important to realise that this deformation is the stationary solution of the elasticity equations. Here, the inertia term $\rho\ddot{u}$ of the elastic half-space is neglected. Hence, the resulting deformation as function of space and time is the solution of the quasi-static elasticity equations.

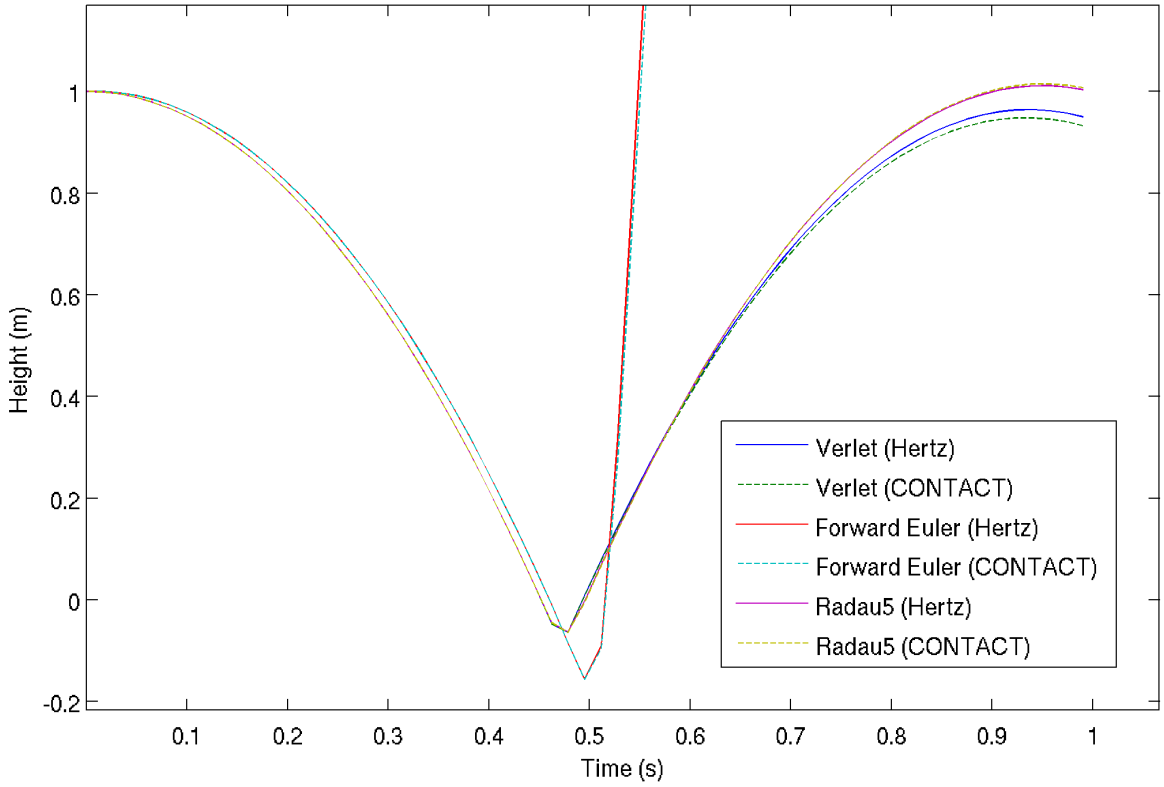


Figure 6.4: Comparison of the exact Hertz solution and the CONTACT solution using the Verlet, Forward Euler, and the Radau5 numerical integration scheme.

We denote the contact force as function of the height z by $F_n(z)$. Note that for $z \geq 0$ the

surface is still in its undeformed state, hence $F_n(z) = 0$. Similarly to the derivation of the differential equation (6.3), we arrive the differential equation

$$\ddot{z} = \frac{F_n(z)}{m} - g \tag{6.11}$$

The same numerical methods can be applied as before. Note that the computation of $F_n(z)$ is relatively expensive (unless of course $z \geq 0$, so that there is no penetration and thus $F_n(z) = 0$). For implicit methods we apply the Picard approach as shown before. This usually requires roughly 5 to 10 iterations before it converges. Hence the total integration 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 static displacement of the surface outside of the contact area can be computed with CONTACT.

We programmed the explicit Verlet, the Forward Euler method, as well as the implicit Radau5 method in Matlab. To compute the normal force, both Hertz theory as well as the CONTACT approach have been implemented. See also Appendix A.2 as well as Appendix A.1 for the Matlab code.

See Figure 6.4 for a comparison between the CONTACT method and the Hertz solution. As one can see, applying Hertz theory will roughly yield the same result as using CONTACT to compute the contact force. A very small difference between the Hertz and CONTACT solution using the Verlet method can be seen, this is most due to to observation we have made in Section 4.6; the normal force computed using CONTACT will be slightly different than the normal force according to Hertz theory, especially if the approach is large. What is interesting is that the Verlet integration scheme behaves much better than the Forward Euler scheme; even though both schemes are of order $\mathcal{O}(\Delta t)$ accurate and require a single function evaluation per time step, the total amount of energy seems to remain nearly constant for the Verlet scheme. We have discussed this phenomenon in Chapter 5. The Radau5 method is of $\mathcal{O}(\Delta t^5)$ accurate so it is not surprising that the this scheme yields the best results.

By using CONTACT to compute the normal force, the deformation of the half-space is also computed at each time step. This deformation is quasi-static; the half-space instantaneously gains speed at the moment of impact. This deformation just after the impact can be seen in Figure 6.5.

6.5 Conclusion

We shave shown how a very basic one-dimensional second order ordinary differential equation for the rigid height of a falling sphere can be derived using Hertz theory. Multiple time integration schemes have been applied to solve this differential equation. Explicit time integration schemes such as Forward Euler tend to become unstable. The amplification of Forward Euler is of course well known and understood, but we have seen that especially during contact the total amount of energy increases significantly. An exception is the Verlet method, which is very energy conserving. The disadvantage, however, is that it is only of order $\mathcal{O}(\Delta t)$ accurate. Implicit methods turned out to be most stable integration schemes. In particular, we favour higher order implicit schemes such as the Radau5 method, due to their stability, energy conservation, and convergence rate.

The same problem has also been solved using the results of CONTACT instead of Hertz theory. By specifying the penetration (which is equal to $\delta(x, y, t) = h(x, y) + z(t)$) CONTACT is capable of computing the normal force F_n. Additionally, this approach resulted in the quasi-static deformation of the elastic half-plane.

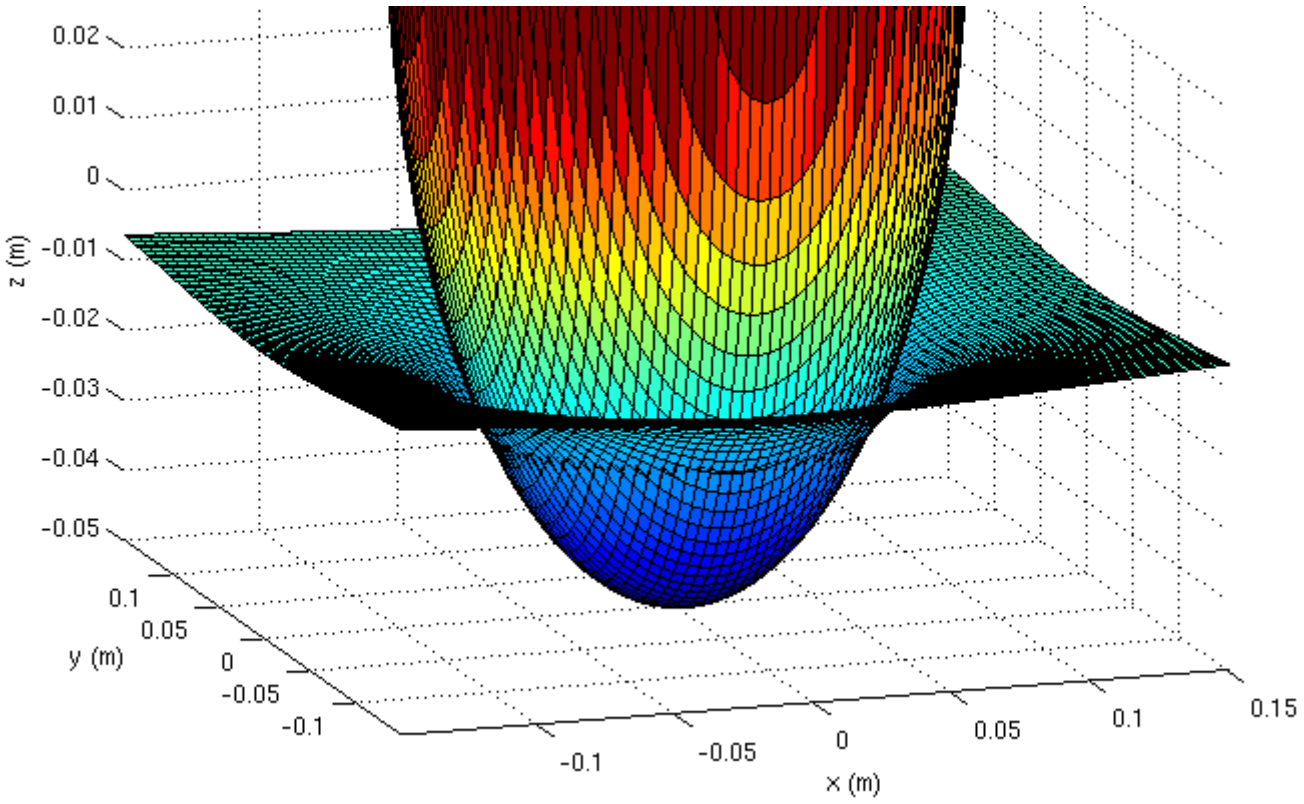


Figure 6.5: The quasi-static deformation of the surface at a certain point in time, computed using CONTACT.

Chapter 7

Finite difference discretisation of the elasticity equations

The previous chapter illustrated the basics of dynamical contact problems involving the computation of the approach as function of time. In most contact problems, one is not only interested in the approach of the two contact problems, but also on the dynamic change of shape of both objects.

In Section 6.4 we have seen that using CONTACT combined with a time integration scheme to solve (6.3) also results in the deformation of both objects as function of time and space. However, this deformation is computed by solving the quasi-static elasticity equations, i.e. the inertia term $\rho\ddot{u}$ is ignored. Hence, deformation occurs instantaneously in that model.

In reality, deformation does not occur instantaneously. Consider a moment t right after the moment the sphere touches the elastic half-plane. At an arbitrary point (x, y) in the contact area, a pressure $p(x, y, t)$ pointing downwards will be exerted at the top surface of the half-plane. Because of the boundary condition for the traction $\mathbf{r} = \sigma \cdot \mathbf{n}$, we in particular have $\sigma_{zz}(x, y, 0, t) = -p(x, y, t)$. This pressure that has arisen at the boundary, however, propagates at a finite speed because of the wave-like nature of the elasticity equation. Hence for each $\delta > 0$ we have $\sigma_{zz}(x, y, -\varepsilon, t + \delta) = 0$ for some small $\varepsilon > 0$. Hence, the partial derivative $\frac{\partial \sigma_{zz}}{\partial z}$ is a very large negative number near the boundary, and 0 everywhere else. As a result,

$$\frac{\partial^2 u_z}{\partial t^2} = \frac{1}{\rho}\left[\frac{\partial \sigma_{zx}}{\partial x} + \frac{\partial \sigma_{zy}}{\partial y} + \frac{\partial \sigma_{zz}}{\partial x} + F_i\right] \tag{7.1}$$

is a very large negative number for boundary elements at the contact area.

The result of this observation is that the surface of the half-space almost instantly gains speed at the moment of impact between the sphere and half-space. This phenomenon happens at a very small time scale. Computing the deformation using the quasi-static approach we have discussed in Section 6.4 therefore seems reasonable. The quasi-static approach does ignore the waves that occur, though.

In this chapter, we will not ignore the inertia and try to solve a very similar problem accurately by using a finite difference discretisation for the system of equations (3.13). A cylindrical

rigid object will be dropped upon an elastic half-space and we are interested in computing the deformation that occurs, especially the behaviour near the contacting boundaries.

Note that in this chapter, we only consider the deformation of a half-plane which happens solely around the contact area. Deformation that happens on a global scale will be discussed in Chapter 8. The finite element method can also be applied to solve (3.13) instead of the finite difference method. This would be a good alternative if the contacting bodies in undeformed state have more complex shapes, i.e. we do not consider a perfect sphere/cylinder or a half-space.

7.1 Formulation of the continuous problem

7.1.1 The problem

Consider a solid cylindrical object of length L and radius R that drops on an elastic cubical body. The body is very large in all directions so that it represents a half-space but remains finite. At all times, the y-axis is parallel to the axis of the cylinder. The cylinder can only move in the z-direction, which is parallel to the direction of gravity. A positive value of z corresponds to a positive height, a negative z means that there is penetration at the surface.

Note that for this 3-dimensional problem, the height of the surface of the cylinder is only a function of x and time t. The y direction is not of importance for the problem at all. The deformation as function of y (with x and z fixed) will be constant. Hence, we consider the problem to be only two dimensional, ignoring the y direction from now on.

We suppose that the elastic body in its undeformed state is a rectangle of width h_x and height h_z. The cylinder (or now circle in 2 dimensions) is centered at $x = 0$. We define our domain Ω to be equal to

$$\Omega = \{(x, z) : -h_x/2 < x < h_x/2, -h_z < z < 0\} \tag{7.2}$$

In this domain, which represents the object in its undeformed state, the elasticity equations hold. The deformation $u_x(x, z, t)$ and $u_z(x, z, t)$ represent the translation of the particle at position (x, z) in the undeformed object after time t.

Furthermore, we define the four boundaries of the domain:

$$\begin{aligned}
\Gamma_1 &= \{(x, 0) | -h_x/2 \leq x \leq h_x/2\} && \text{(the top boundary)} \\
\Gamma_2 &= \{(-h_x/2, z) | -h_z \leq z \leq 0\} && \text{(the left boundary)} \\
\Gamma_3 &= \{(h_x/2, z) | -h_z \leq z \leq 0\} && \text{(the right boundary), and} \\
\Gamma_4 &= \{(x, -h_z) | -h_x/2 \leq x \leq h_x/2\} && \text{(the bottom boundary)}
\end{aligned} \tag{7.3}$$

See Figure 7.1 for an illustration.

For each boundary, different boundary conditions can be described. We assume that the object is fixed at the left, right and bottom boundaries, i.e. the particles at these boundaries stay in their position. This corresponds to the boundary condition $u_x(\mathbf{x}, t) = u_z(\mathbf{x}, t) = 0$ for all $\mathbf{x} = (x, z) \in \Gamma_2 \cup \Gamma_3 \cup \Gamma_4$. The cylinder will touch the surface at the top boundary Γ_1, hence we will describe boundary conditions for the stress at this boundary. The pressure at Γ_1 will be computed using Hertz theory or can be computed using CONTACT.

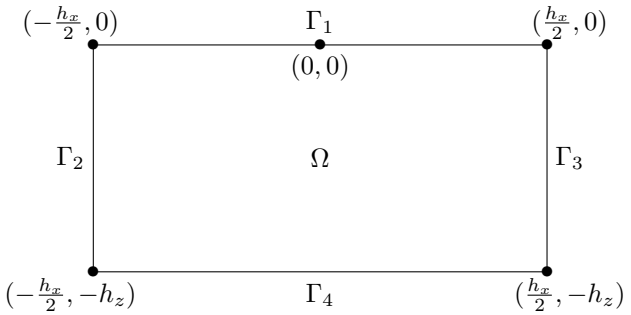


Figure 7.1: The domain Ω and its boundaries.

7.1.2 Computation of the top boundary condition using Hertz theory

Let $z_c(t)$ be the height of the cylinder, measured from its lowest point at $x = 0$ (so the axis of the cylinder has height $z_c(t) + R$). The approach is therefore $\delta(t) = -z_c(t)$. Then, by Hertz theory, the maximum pressure is equal to

$$p_0 = \frac{2}{\pi} E^* \sqrt{\frac{\delta(t)}{R}} \tag{7.4}$$

assuming that $\delta(t) \geq 0$, i.e. there is penetration. Here, $E^* = \frac{E}{1-\nu^2}$ with E and ν being the elastic and shear modulus of the surface, respectively. Next, the contact radius a is computed. This is simply equal to $a = \sqrt{R^2 - (R - \delta(t))^2}$ by the Pythagoras theorem.

Hertz theory now describes the pressure distribution p:

$$p(x) = \begin{cases} p_0 \left(1 - \dfrac{|x|}{a^2}\right)^{1/2} & \text{for } |x| \leq a \\ 0 & \text{for } |x| > a \end{cases} \tag{7.5}$$

This pressure distribution is then set as our boundary condition $\sigma \mathbf{n} = [0; -p]$, which corresponds in the 2D case to the boundary conditions $\sigma_{13} = 0$ and $\sigma_{33} = -p$.

As shown before, the contact force per unit width is given by

$$F_c = \frac{4}{3} E^* R^{1/2} \max\{\delta, 0\}^{3/2} \tag{7.6}$$

And therefore, by Newtons second law, the height z of the cylinder must satisfy the following differential equation:

$$\ddot{z_c} = \frac{F_c}{m} - g \tag{7.7}$$

where m is the mass of the cylinder per unit width, which is equal to $m = \pi R^2 \rho$, where ρ is the density of the cylinder.

7.1.3 Formulation of the differential algebraic equation

Using the pressure distribution p for our boundary condition at Γ_1, we can describe the mathematical formulation of the problem. As described in Chapter 3, the linear elasticity equations can we written as

$$G\frac{\partial^2 u_q}{\partial x_r^2} + (\lambda + G)\frac{\partial^2 u_r}{\partial x_q x_r} + F_q = \rho\frac{\partial^2 u_q}{\partial t^2} \qquad \text{for } q, r = 1, 3 \tag{7.8}$$

where $u_1 = u_x$ is the deformation in the x-direction and $u_z = u_3$ the deformation in the z direction. In this formula, both the stress and the strain components are eliminated. This formulation, however, is not recommended since the boundary condition at Γ_1 is described using σ_{13} and σ_{33}, which were eliminated from the differential equation.

Therefore, we will instead take a look at the elasticity equations in their usual form. Combined with the boundary and initial conditions, we get

$$\begin{aligned}
\rho\frac{\partial^2 u_q}{\partial t^2} &= \frac{\partial \sigma_{qn}}{\partial x_n} + F_q && \text{for } (x,z) \in \Omega, t > 0, q = 1, 3 \\
e_{qr} &= \frac{1}{2}\left(\frac{\partial u_q}{\partial x_r} + \frac{\partial u_r}{\partial x_q}\right) && \text{for } (x,z) \in \Omega, t \geq 0, q, r = 1, 3 \\
\sigma_{qr} &= \lambda e_{nn}\delta_{qr} + 2Ge_{qr} && \text{for } (x,z) \in \overline{\Omega}, t \geq 0, q, r = 1, 3 \\
u_x = u_y &= 0 && \text{for } (x,z) \in \Gamma_2 \cup \Gamma_4 \cup \Gamma_4, t \geq 0 \\
\sigma_{13} &= 0 && \text{for } (x,z) \in \Gamma_1, t \geq 0 \\
\sigma_{33} &= -p(x, z_c) && \text{for } (x,z) \in \Gamma_1, t \geq 0 \\
u_x(x,z,0) = u_z(x,z,0) &= 0 && \text{for } (x,y) \in \overline{\Omega}
\end{aligned} \tag{7.9}$$

where $\overline{\Omega} = \Omega \cup (\Gamma_1 \cup \Gamma_2 \cup \Gamma_3 \cup \Gamma_4) = \{(x,z) : -h_x/2 \leq x \leq h_x/2, -h_z \leq z \leq 0\}$. Einstein convention is used for the n parameter. This system is called a differential algebraic equation, because some parts are differential equations while others are algebraic equalities. In this case, the first equation contains a time-derivative, the second and third do not. The last two equations are instantaneous and must be satisfied at all times.

Furthermore, we also have the ordinary differential equation for the height of the ball. At $t = 0$, the ball has a height z_0 and has no vertical speed.

$$\begin{aligned}
\ddot{z}_c &= \frac{4}{3m}E^* R^{1/2}\max\{0, -z_c\}^{3/2} - g \quad \text{for } t > 0 \\
z_c(0) &= z_0 \\
\frac{\mathrm{d}z_c}{\mathrm{d}t}(0) &= 0
\end{aligned} \tag{7.10}$$

Note that differential algebraic equation (7.9) and differential equation (7.10) are connected, since the boundary condition at Γ_1 depends on the height of the ball.

7.2 Discretisation

If the geometry of the object in its undeformed state is not too complicated, the problem can be discretised by using the finite difference method. When used together with a good time integration scheme, this will result in a good simulation of the deformation of the object.

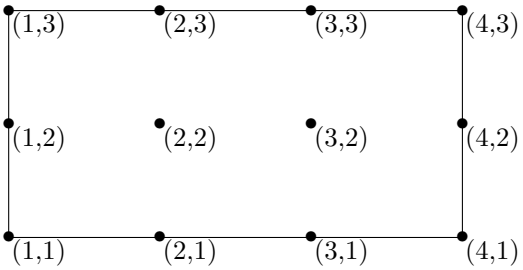


Figure 7.2: The discretisation of Ω.

For our problem, the domain of computation is a rectangle. There are many ways to discretise this domain. We will consider a straightforward discretisation; a uniform grid where all components ($u_x, u_z, \sigma_{11}, \sigma_{13}, \sigma_{33}, e_{11}, e_{13}$, and e_{33}) are defined on the same grid elements. The disadvantage of this approach is that the accuracy might be lower at the boundary. Another option is to use a staggered approach, where the grid elements of different components can be defined at different positions in the domain.

The uniform grid has m_x and m_z elements in the x and z direction. At each row and column, we start numbering at 1, which is the element at the left or bottom boundary. The m_x^{th} and m_z^{th} element is located at the right and top boundary, respectively. Therefore, we have grid width $\Delta x = \frac{h_x}{m_x - 1}$ and height $\Delta z = \frac{h_z}{m_z - 1}$. The position p of each grid point (i, j) has coordinates

$$\begin{aligned} (p(i,j))_x &= i\Delta x - \frac{h_x}{2} \\ (p(i,j))_z &= j\Delta z - h_z \end{aligned} \tag{7.11}$$

As an example, suppose that $h_x = 6, h_z = 3, m_x = 4$, and $m_z = 3$. Then $\Delta x = \frac{6}{4-1} = 2$ and $\Delta z = \frac{3}{3-1} = \frac{3}{2}$. See figure 7.2 for an illustration.

Note that the usual global ordering is used to map a grid point (i, j) to point I, where $1 \leq I \leq m = m_x m_z$, so that the example in Figure 7.3 can be represented using global coordinates by

Suppose the Forward (or Backward) Euler method is chosen as our integration scheme. The first step is to rewrite (7.9) as a system of first order derivatives. Define $v = \frac{\partial u}{\partial t}$, then we have differential equations

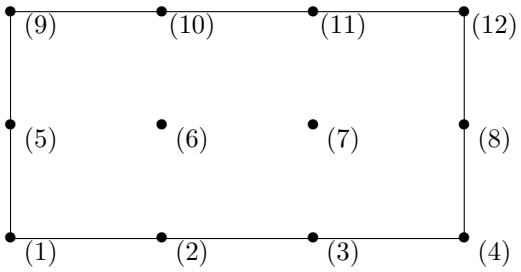


Figure 7.3: The discretisation of Ω using a global index.

$$\begin{aligned}
\frac{\partial u_q}{\partial t} &= v_q && \text{for } (x,z) \in \Omega, t > 0, q = 1,3 \\
e_{qr} &= \frac{1}{2}\left(\frac{\partial u_q}{\partial x_r} + \frac{\partial u_r}{\partial x_q}\right) && \text{for } (x,z) \in \Omega, t > 0, q, r = 1,3 \\
\sigma_{qr} &= \lambda e_{nn}\delta_{qr} + 2Ge_{qr} && \text{for } (x,z) \in \overline{\Omega}, t \geq 0, q, r = 1,3 \\
\frac{\partial v_q}{\partial t} &= \frac{1}{\rho}\left(\frac{\partial \sigma_{qn}}{\partial x_n} + F_q\right) && \text{for } (x,z) \in \Omega, t > 0, q = 1,3
\end{aligned} \tag{7.12}$$

The equality

$$e_{qr} = \frac{1}{2}\left(\frac{\partial u_q}{\partial x_r} + \frac{\partial u_r}{\partial x_q}\right) \tag{7.13}$$

can be discretised using standard central differences. For the $(i,j)^{\text{th}}$ component of the grid, we have

$$\begin{aligned}
(e_{11})_{i,j} &\approx \frac{(u_1)_{i+1,j} - (u_1)_{i-1,j}}{2\Delta x} \\
(e_{13})_{i,j} = (e_{31})_{i,j} &\approx \frac{1}{2}\left(\frac{(u_1)_{i,j+1} - (u_1)_{i,j-1}}{2\Delta z} + \frac{(u_3)_{i+1,j} - (u_3)_{i-1,j}}{2\Delta x}\right) \\
(e_{33})_{i,j} &\approx \frac{(u_3)_{i,j+1} - (u_3)_{i,j-1}}{2\Delta z}
\end{aligned} \tag{7.14}$$

These approximations are valid and of second order accuracy for interior points (i,j). For the boundaries, the formulation is no longer valid. A possible fix for this is to use one-sided differences instead. For example, at the left boundary we can use

$$(e_{11})_{1,j} \approx \frac{(u_1)_{2,j} - (u_1)_{1,j}}{\Delta x} \tag{7.15}$$

However, this is only of first order accuracy. Usually, this is not a problem if the discretisation at internal grid points has a higher order of accuracy. However, for better accuracy, a higher order one-sided difference scheme can be used instead. As an example, for the left boundary we can use the numerical approximation

$$(e_{11})_{1,j} \approx \frac{-3(u_1)_{1,j} + 4(u_1)_{2,j} - (u_1)_{3,j}}{\Delta x} \tag{7.16}$$

which is an approximation of $(e_{11})_{1,j} = (\frac{\partial u_1}{\partial x_1})_{1,j}$ of second order accuracy. Using this scheme for the boundary points and central differences for the internal points, the total discretisation will therefore be of second order accuracy.

The general disadvantage of using one-sided difference schemes is that the corresponding discretisation matrix will no longer be symmetric. Hence, Cholesky decomposition and other tricks that rely on the symmetry of the matrix can no longer be used to solve the linear system. However, the matrix corresponding to equation (7.9) is not symmetric as we will see shortly, regardless of the symmetry of A. A one-sided difference scheme at the boundary can therefore easily be implemented without causing more problems.

These equations for $(e_{11})_{i,j}$, $(e_{13})_{i,j}$, and $(e_{33})_{i,j}$ for $1 \leq i \leq m_x, 1 \leq j \leq m_z$ form a system. This system is linear and can therefore be written as $\mathbf{e} = B\mathbf{u}$. In this expression, B is the matrix that contains the central (and one-sided at the boundary) differences coefficients, and $\mathbf{e} = [\mathbf{e}_{11}; \mathbf{e}_{13}; \mathbf{e}_{33}]$ is the vector that contains all the strain components.

The part

$$\sigma_{qr} = \lambda e_{nn}\delta_{qr} + 2Ge_{qr} \tag{7.17}$$

can directly be written in vector notation as $\sigma = C\mathbf{e}$. σ is the vector that contains all the stress components: $\sigma = [\sigma_{11}; \sigma_{13}; \sigma_{33}]$.

Importantly, the equation

$$\frac{\partial u_q}{\partial t} = (v_q)_t, \qquad q = 1, 3 \tag{7.18}$$

is only valid for internal grid points (i, j) since the elasticity equations and therefore also the discretised equations (7.14) are only valid in Ω. Let

$$\Omega_{\text{in}} = \{(i, j), 1 \leq i \leq m_x, 1 \leq j \leq m_z, (i, j) \text{ is an internal grid point}\} \tag{7.19}$$

be the set of internal grid points. We let m_{in} denote the amount of internal grid points. Then equation (7.18) corresponds to $\frac{\mathrm{d}}{\mathrm{d}t}(u_q)_{i,j} = (v_q)_{i,j}$ for $(i, j) \in \Omega_{\text{in}}$. Define $\mathring{\mathbf{u}}_1$ and $\mathring{\mathbf{u}}_3$ be the vectors of length m_{in} that contain the internal components of $\mathbf{u}_1$ and $\mathbf{u}_3$, respectively. We do the same to define $\mathring{\mathbf{v}}_1$ and $\mathring{\mathbf{v}}_3$. Then (7.18) can simply be written as

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathring{\mathbf{u}} = \mathring{\mathbf{v}} \tag{7.20}$$

where $\mathring{\mathbf{u}} = [\mathring{\mathbf{u}}_1; \mathring{\mathbf{u}}_3]$ and $\mathring{\mathbf{v}} = [\mathring{\mathbf{v}}_1; \mathring{\mathbf{v}}_3]$.

Lastly, the part

$$\frac{\partial v_q}{\partial t} = \frac{1}{\rho}\left(\frac{\partial \sigma_{qn}}{\partial x_n} + F_q\right) \tag{7.21}$$

is also only valid for internal grid points. Using central differences as before, one can approximate this term:

$$\begin{aligned}
\frac{\partial}{\partial t}(v_1)_{i,j} &\approx \frac{1}{\rho}\left[\frac{(\sigma_{11})_{i+1,j} - (\sigma_{11})_{i-1,j}}{2\Delta x} + \frac{(\sigma_{13})_{i,j+1} - (\sigma_{13})_{i,j-1}}{2\Delta z} + F_1(i, j)\right] \\
\frac{\partial}{\partial t}(v_3)_{i,j} &\approx \frac{1}{\rho}\left[\frac{(\sigma_{31})_{i+1,j} - (\sigma_{31})_{i-1,j}}{2\Delta x} + \frac{(\sigma_{33})_{i,j+1} - (\sigma_{33})_{i,j-1}}{2\Delta z} + F_3(i, j)\right]
\end{aligned} \tag{7.22}$$

Let $\mathring{\mathbf{v}} = [\mathring{\mathbf{v}}_1; \mathring{\mathbf{v}}_3]$ and $\mathbf{F} = [\mathbf{F}_1; \mathbf{F}_3]$. Then this can be written as $\frac{\mathrm{d}}{\mathrm{d}t}\mathring{\mathbf{v}} = T\sigma + \mathbf{F}$.

Hence, the discretised differential algebraic equation can be written as

$$\begin{cases} \dfrac{\mathrm{d}}{\mathrm{d}t}\mathring{\mathbf{u}}^k = \mathring{\mathbf{v}}^k \\ \quad \mathbf{e}^k = B\mathbf{u}^k \\ \quad \sigma^k = C\mathbf{e}^k \\ \dfrac{\mathrm{d}}{\mathrm{d}t}\mathring{\mathbf{v}}^k = T\sigma^k + \mathbf{F}^k \\ \quad +\text{boundary conditions} \end{cases} \tag{7.23}$$

For each grid point at the boundary, two boundary conditions should be given for a total of $2(m - m_{\text{in}})$ conditions.

7.2.1 Implementation with the Forward Euler approach

The Forward Euler method is applied by setting

$$\begin{aligned} \mathring{\mathbf{u}}^{k+1} &= \mathring{\mathbf{u}}^k + \Delta t \mathring{\mathbf{v}}^k \\ \mathring{\mathbf{v}}^{k+1} &= \mathring{\mathbf{v}}^k + \Delta t(T\sigma^k + \mathbf{F}^k) \end{aligned} \tag{7.24}$$

By combining everything, we arrive at the following linear system:

$$\begin{pmatrix} I & 0 & 0 & 0 \\ -B & I & 0 & 0 \\ 0 & -C & I & 0 \\ 0 & 0 & 0 & I \end{pmatrix} \begin{pmatrix} \mathbf{u} \\ \mathbf{e} \\ \sigma \\ \mathring{\mathbf{v}} \end{pmatrix}^{k+1} = \begin{pmatrix} \mathbf{u}^k + \Delta t \mathbf{v}^k \\ \mathbf{0} \\ \mathbf{0} \\ \mathring{\mathbf{v}}^k + \Delta t(T\sigma^k + \mathbf{F}^k) \end{pmatrix} \tag{7.25}$$

which we will notate as $A\mathbf{x}^{k+1} = \mathbf{f}^k$.

Note that the identity matrix at the first block row is only described for internal grid points (i, j). For boundary points, the diagonal element at the corresponding row is 0 by default. Boundary conditions should be added at this row.

To illustrate this, consider a similar 1D situation where $m_z = 4$. Then $\Omega_{\text{in}} = \{2, 3\}$. Suppose that the stress at the boundaries is given and equal to 1 at the left boundary and -1 at the right boundary. Then the first four rows of the equation are

$$\begin{pmatrix} 0 & 0 & 0 & 0 & \dots & 1 & 0 & 0 & 0 & \dots \\ 0 & 1 & 0 & 0 & \dots & 0 & 0 & 0 & 0 & \dots \\ 0 & 0 & 1 & 0 & \dots & 0 & 0 & 0 & 0 & \dots \\ 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 & 1 & \dots \end{pmatrix} \mathbf{x}^{k+1} = \begin{pmatrix} 1 \\ u_2^k + \Delta t v_2^k \\ u_3^k + \Delta t v_3^k \\ -1 \end{pmatrix} \tag{7.26}$$

The first row here simply sets the stress $(\sigma_{33}^{k+1})_1$ equal to 1. The second and third row represents the time integration for elements u_2^k and u_3^k. Lastly, the fourth row makes sure that $(\sigma_{33}^{k+1})_4$ is equal to -1.

7.2.2 Implementation with the Backward Euler approach

The Backward Euler method is very similar:

$$\begin{aligned} \mathring{\mathbf{u}}^{k+1} &= \mathring{\mathbf{u}}^{k} + \Delta t \mathring{\mathbf{v}}^{k+1} \\ \mathring{\mathbf{v}}^{k+1} &= \mathring{\mathbf{v}}^{k} + \Delta t (T\sigma^{k+1} + \mathbf{F}^{k+1}) \end{aligned} \tag{7.27}$$

By moving the $\mathring{\mathbf{v}}^{k+1}$ and σ^{k+1} to the left side, the following linear system can be derived:

$$\begin{pmatrix} I & 0 & 0 & -\Delta t I \\ -B & I & 0 & 0 \\ 0 & -C & I & 0 \\ 0 & 0 & -\Delta t T & I \end{pmatrix} \begin{pmatrix} \mathbf{u} \\ \mathbf{e} \\ \sigma \\ \mathring{\mathbf{v}} \end{pmatrix}^{k+1} = \begin{pmatrix} \mathbf{u} \\ \mathbf{e} \\ \sigma \\ \mathring{\mathbf{v}} \end{pmatrix}^{k} + \Delta t \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathring{\mathbf{v}}^{k} + \Delta t \mathbf{F}^{k+1} \end{pmatrix} \tag{7.28}$$

or $A\mathbf{x}^{k+1} = \mathbf{x}^{k} + \Delta t \mathbf{f}^{k+1}$. If, furthermore, the body force $\mathbf{F}$ is constant (zero in many cases), the right hand side does not depend on time step $k+1$. In that case, the solution of the linear system can be computed directly and is equal to $\mathbf{x}^{k+1} = A^{-1}(\mathbf{x}^{k} + \Delta t \mathbf{f}^{k})$.

7.3 Dynamic boundary conditions

Since the boundary condition at Γ_1 depends on the height of the cylinder, (7.9) and (7.10) should be solved at the same time.

As an example, for Backward Euler we have

$$\begin{aligned} z_c^{k+1} &= z_c^{k} + \Delta t v_c^{k+1} \\ v_c^{k+1} &= v_c^{k} + \Delta t \left(\frac{4}{3m} E^* R^{1/2} \max\{0, -z_c^{k+1}\}^{3/2} - g \right) \end{aligned} \tag{7.29}$$

This can be written as

$$\begin{pmatrix} I & 0 & 0 & -\Delta t I & 0 & 0 \\ -B & I & 0 & 0 & 0 & 0 \\ 0 & -C & I & 0 & 0 & 0 \\ 0 & 0 & -\Delta t T & I & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -\Delta t \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \mathbf{u} \\ \mathbf{e} \\ \sigma \\ \mathring{\mathbf{v}} \\ z_c \\ v_c \end{pmatrix}^{k+1} = \begin{pmatrix} \mathbf{u} \\ \mathbf{e} \\ \sigma \\ \mathring{\mathbf{v}} \\ z_c \\ v_c \end{pmatrix}^{k} + \Delta t \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ 0 \\ \frac{4}{3m} E^* R^{1/2} \max\{0, -z_c^{k+1}\}^{3/2} - g \end{pmatrix} \tag{7.30}$$

or $A\mathbf{x}^{k+1} = \mathbf{x}^{k} + \mathbf{f}^{k+1}$. The only non-linear term is contained in the last equation. We know that if $z^{k+1} \geq 0$, i.e. the ball will not touch the surface at time t^{k+1}, then the linearity of (7.30) drops out of the system. In that case,

$$\begin{pmatrix} \mathbf{u} \\ \mathbf{e} \\ \sigma \\ \mathring{\mathbf{v}} \\ z_c \\ v_c \end{pmatrix}^{k+1} = \begin{pmatrix} I & 0 & 0 & -\Delta t I & 0 & 0 \\ -B & I & 0 & 0 & 0 & 0 \\ 0 & -C & I & 0 & 0 & 0 \\ 0 & 0 & -\Delta t T & I & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -\Delta t \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}^{-1} \begin{pmatrix} \mathbf{u}^k \\ \mathbf{0} \\ \mathbf{0} \\ \mathring{\mathbf{v}}^k \\ z_c^k \\ v_c^k - \Delta t g \end{pmatrix} \tag{7.31}$$

Difficulties occur when $z^{k+1} < 0$, i.e. there is penetration. $\mathbf{x}^{k+1}$ can then no longer be explicitly stated as function of $\mathbf{x}^k$. It is possible to use the height z_c^k instead of z_c^{k+1} for (7.30). However, this changes the time integration scheme and causes stability issues. Therefore, one should use a implicit solver for this system. This deals with both the non-linearity of (7.29) and the not yet known right hand vector $\mathbf{f}^{k+1}$.

The following algorithm is an example on how to apply Picard iteration for this problem. Let $\mathbf{x}^k = [\mathbf{u}^k; \mathbf{e}^k; \sigma^k; \mathbf{v}^k; z_c^k; v_c^k]$.

Algorithm 7.1 Applying the Backward Euler method using Picard iteration

1: Compute initial condition $\mathbf{x}^0$.
2: **for** $k = 1$ to n **do** ▷ Loop over each of n time steps
3: Set $\mathbf{x} = \mathbf{x}^{k-1}$ ▷ Initial guess at each time step
4: **repeat**
5: Compute vector $\mathbf{f}(\mathbf{x})$ as in (7.30).
6: Solve $A\mathbf{x}^* = \mathbf{f}(\mathbf{x})$ for $\mathbf{x}^*$.
7: Set error $= \|\mathbf{x}^* - \mathbf{x}\|$ for $\mathbf{x}^*$.
8: Set $\mathbf{x} = \mathbf{x}^*$
9: **until** error $\leq \varepsilon$
10: Set $\mathbf{x}^k = \mathbf{x}$.
11: **end for**

Here, n is the number of iterations. If this method converges, then $A\mathbf{x}^{k+1} = \mathbf{f} \approx \mathbf{f}^{k+1}$ so that we have approximated the solution of the non-linear implicit system.

7.4 Numerical results

The finite difference discretisation of the elasticity equation with dynamic boundary conditions has been programmed in Matlab. Both the Forward Euler as well as the backward Euler have been applied. For the Backward Euler method, algorithm (7.1) has been applied. See Appendix A.3 for the Matlab implementation.

A uniform mesh with $m_x = 40$ and $m_y = 30$ elements has been used to discretise the half-plane. This matrix A as in (7.30) for this very small problem is already roughly 12000×12000 large. In generic 3D situation this would be even larger. As a result, computing solution of $A\mathbf{x} = \mathbf{v}$ is computationally very expensive. This is even worse for implicit methods that require the use of an iterative process at each time step. If CONTACT is used to compute the pressure instead of applying Hertz theory, computing the right hand side $\mathbf{f}$ would also require much more computational power. In short, we are severely restricted on the amount of mesh points.

In the implementation we have used to different values of Δt. We use $\Delta t = 10^{-3}$ for those time steps j where $z_c^{j-1} > 0$, i.e. there is no penetration yet, hence we can solve (7.31) instead of the implicit (7.30). If $z_c^{j-1} > 0$, i.e. there was penetration at the previous time step, we use $\Delta t = 10^{-4}$. There are two reasons for the dynamic time step. First of all, contact happens very quickly and the contact force and pressure therefore change rapidly as function of the time. It is therefore important to observe this in great detail. Additionally, for implicit methods, the Picard iteration can break down if either the time step or the Young's modulus of the material is too large.

In Figure 7.4 the deformation of the half-plane can be seen just right after impact. Backward Euler has been applied to get these results. The impact of the sphere can clearly be seen and the boundary of the half-plane follows the boundary of the cylinder perfectly. During the impact, waves appear near the boundary which slowly propagate to the rest of the half-plane.

There are some problems that occur, though. As mentioned before, the algorithm is computationally extremely expensive if a fine mesh for the elastic half-plane is used. The accuracy of the discretisation is therefore very limited. Furthermore, approximately halfway during impact a gap between the half-plane and the cylinder emerges when the elements at the boundary of the half-plane continue to move downwards but the cylinder has reached its lowest point. This is an unrealistic phenomenon and might point to a problem in the formulation of the pressure at the boundary.

Another major problem is that if the materials is made of a stiff material like steel, the iterative process never converges. A similar problem occurs in Chapter 10. There a solution is proposed which might also be applied on this problem.

The explicit Forward Euler turned out to be unstable. Right after the impact between the cylinder and the half-plane occurs, the elements at the boundary of the half-plane jump to random directions. This instablity starts at the contact area and slowly propagates to the rest of the half-plane. This phenomenon can be seen in Figure 7.5.

It appears that there is a hidden CFL condition needs to be satisfied in order for the explicit time integration to be stable. This will be discussed for a similar differential equation in Section 8.3.3.

7.5 Conclusion

In this chapter we have shown how the finite difference method can be used in order to discretise the elasticity equations. Central differences (except near the boundary) can be used to approximate the spatial derivatives. To solve the differential algebraic equation, a time integration scheme has been combined with the algebraic relations between the displacement and the strain as well as the relations between the stress and the strain into one single system.

A similar problem as in Chapter 6 involving a cylinder falling on an elastic half-space has been solved using this approach. Explicit time integration schemes break down quickly, but implicit schemes like Backward Euler combined with a Picard approach can result in fairly realistic deformations. However, this approach is computationally extremely expensive, can result in unrealistic scenarios, or even breaks down in the iterative process for rigid materials like steel.

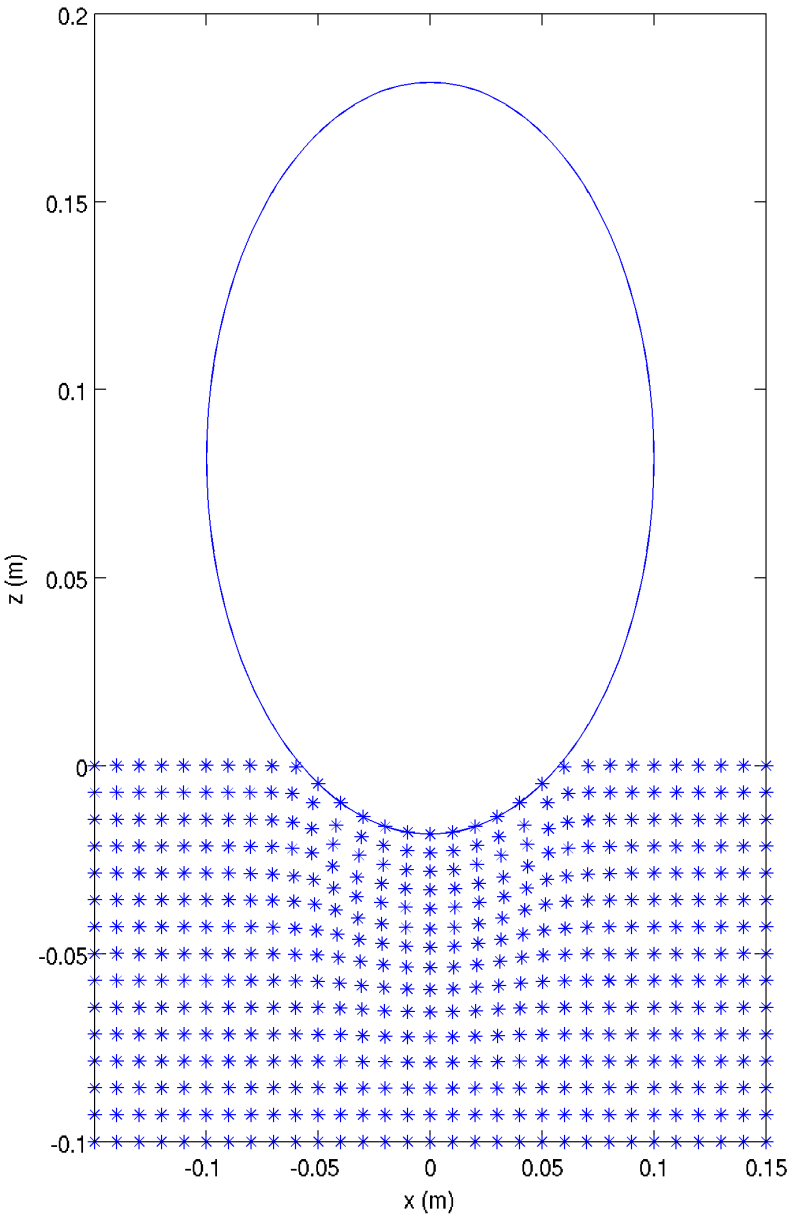


Figure 7.4: The deformation of the surface just after the impact of a rigid cylinder, using Backward Euler as integration scheme.

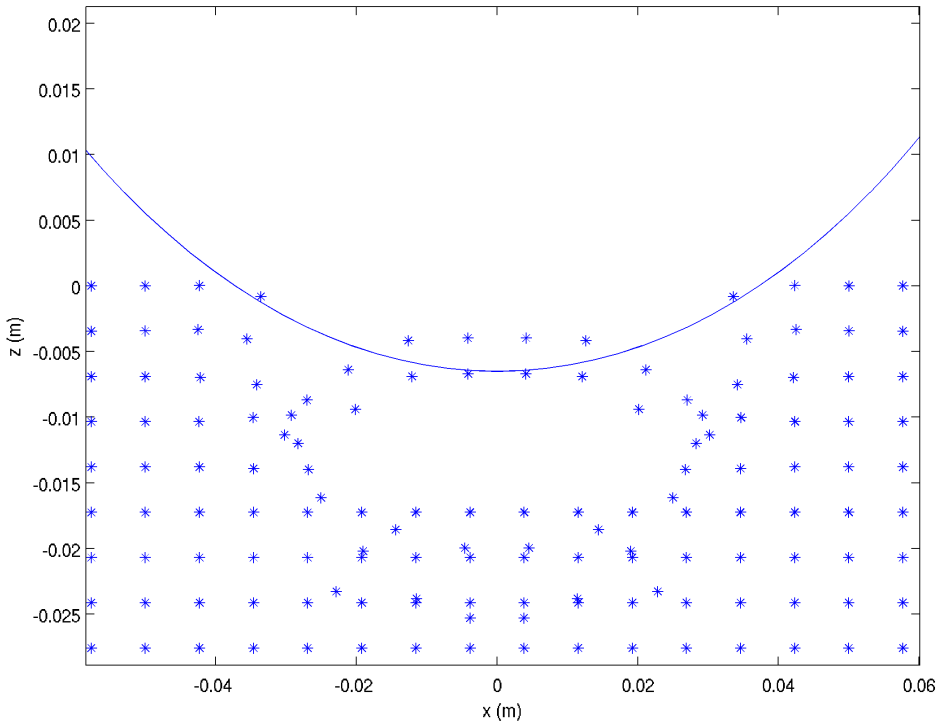


Figure 7.5: The deformation of the surface just after the impact of a rigid cylinder using the instable Forward Euler as integration scheme.

Chapter 8

Global deformation of a bridge

In the previous chapters, we have discussed the contact dynamic of an object falling on an elastic half-plane. The half-plane itself is fixed and does not move as a whole. Deformation only occurs strictly around the contact area. This is what we call local deformation.

In this chapter, we will discuss the global deformation of a bridge. Consider an elastic bridge of length L with an object exerting a force downwards. Not only will this cause local deformation of the beam around the initial point of contact as we have discussed previously, the whole beam will also deform as a whole. This is what we will describe as global deformation.

The difference between this situation and the problem of the same object exerting a force on an half-plane is hidden in the boundary condition at the bottom of the bridge and half-plane. For the half plane, we have the boundary condition $\mathbf{0} = \lim_{y \to -\infty} \mathbf{u}(x, y, t)$ for all $x \in \mathbb{R}$ and $t \geq 0$. That is, particles far away from the contact area are not capable of moving. Particles at the bottom of a bridge, however, are not fixed to anything. But we do know that there is no external force at the bottom of the bridge. This can be translated to the traction boundary condition $\sigma \cdot \mathbf{n} = \mathbf{0}$.

In this chapter, we will focus on the theory of global deformations. In particular, we are interested in solving a moving load problem, such as a train riding on a bridge. The force of the train exerting on the bridge differs with respect to both place and time. The bridge will deform globally as a train moves over it. To compute this deformation, two different approaches will be discussed; the discrete approach and the modal approach.

8.1 The 1D dynamic beam equation

Consider an elastic beam (such as a bridge) and suppose a force is being exerted on the top of this beam. The resulting deformation can be computed by solving the elasticity equations using an approach such as described in Chapter 7.

If this beam is supposed to be long and thin, which is the case if we consider a bridge, Euler-Bernoulli theory also becomes applicable for this problem. The Euler-Bernoulli beam equation is a simplification of the elasticity equations and describes the deformation of a beam as function of space and time. Shear deformation is being ignored 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 [15].

Euler-Bernoulli theory assumes that particles in the beam cannot move in the x direction. Additionally, it is assumed that the vertical displacement is constant for particles at the same x-coordinate; the effect in the z direction is being ignored. As the result, the displacement $u = u_z$ described by the beam equation is only a function of x and t. The beam equation is given by

$$\frac{\partial^2}{\partial x^2}\left(E(x)I(x)\frac{\partial^2 u}{\partial x^2}\right) = -\rho(x)\frac{\partial^2 u}{\partial t^2} + p(x,t) \tag{8.1}$$

Here E is the elasticity modulus of the material of the beam. The parameter $\rho(x)$ 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

$$I(x) = \int_{A(x)} d(z)^2 \mathrm{d}z \tag{8.2}$$

in the one dimensional case, or

$$I(x) = \iint_{A(x)} d(y,z)^2 \mathrm{d}y\mathrm{d}z \tag{8.3}$$

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.

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

$$EI\frac{\partial^4 u}{\partial x^4} = -\rho\frac{\partial^2 u}{\partial t^2} + p(x,t) \tag{8.4}$$

8.2 Boundary and initial conditions

Equation (8.1) is a partial differential equation which is of fourth order with respect to the x variable, hence we need 4 spatial boundary conditions. Various boundary conditions are possible for this problem and the choice depends on the problem. A reasonable assumption (if we see the beam as a bridge) is 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:

$$\begin{aligned} u(0,t) = 0 = \frac{\partial u}{\partial x}(0,t) \\ u(L,t) = 0 = \frac{\partial u}{\partial x}(L,t) \end{aligned} \tag{8.5}$$

These conditions are called clamped boundary conditions. Another (more realistic) possibility is to consider 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. This boundary conditions is satisfied if the beam is free to rotate at the boundaries and does not experience any torque. Usually, this torque is insignificant and may therefore be ignored.

The corresponding boundary conditions for this case are, since both E and I are assumed to be non-zero:

$$\begin{aligned} u(0,t) = 0 = \frac{\partial^2 u}{\partial x^2}(0,t) \\ u(L,t) = 0 = \frac{\partial^2 u}{\partial x^2}(L,t) \end{aligned} \tag{8.6}$$

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. Usually, we assume the beam is in rest at $t = 0$, hence $u(x, 0) = \frac{\partial u}{\partial t}u(x, 0) = 0$ for all $x \in [0, L]$.

8.3 The discrete problem

8.3.1 Discretisation of the differential equation

For a general force function $p(x, t)$, differential equation (8.1) cannot be solved analytically. A straightforward approach to get an approximation of the solution is by discretising differential equation (8.4). Both the Finite Element method and the Finite Difference method can be applied for this problem. As we will discuss in Section 8.4, a modal approach is also very effective.

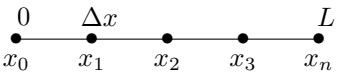


Figure 8.1: The discretisation of the beam for $n = 4$.

The simple geometry of a one dimensional beam allows us to apply the Finite Difference method. In this section we will solely focus on this method, but of course a Finite Element approach can be used instead. 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 (8.4), 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 (8.4) and rearranging the terms a bit, we arrive at the system of equations

$$\begin{aligned} \frac{\partial u}{\partial t} &= v \\ \frac{\partial v}{\partial t} &= -\frac{EI}{\rho}\frac{\partial^4 u}{\partial x^4} + \frac{p(x,t)}{\rho} \end{aligned} \tag{8.7}$$

For convention, we denote u_j^k as a shorthand notation of $u(x_j, t_k)$. As shown in [18], 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:

$$\frac{\partial^4}{\partial x^4} u_j(t) = \frac{u_{j-2}(t) - 4u_{j-1}(t) + 6u_j(t) - 4u_{j+1}(t) + u_{j+2}(t)}{\Delta x^4} + \mathcal{O}(\Delta x^2) \tag{8.8}$$

8.3.2 Implementation of the boundary conditions

The Dirichlet boundary conditions $u(0,t) = u(L,t) = 0$ can be applied by adding the equation $u_0^k = 0$ and $u_n^k = 0$, which corresponds to the element at the boundary $x = 0$ or $x = L$, respectively, to the linear system of equations, similarly to our approach in Chapter 7. Alternatively, one can simply eliminate all the terms u_0^k or u_n^k directly for each expression that contains these terms.

Both the boundary conditions for the clamped beam $\frac{\partial u}{\partial x}(0,t) = \frac{\partial u}{\partial x}(L,t) = 0$ as well as the boundary conditions for the simply supported beam $\frac{\partial^2 u}{\partial x^2}(0,t) = \frac{\partial^2 u}{\partial x^2}(L,t) = 0$ are implemented by introducing a virtual point outside the domain $[0, L]$ and then eliminating it by using the boundary conditions.

Consider the left boundary. We introduce a virtual point $x_{-1} = -\Delta x$. Since at $i = 0$ we added the equation $u_0^k = 0$ to the linear system (or eliminated u_0^k by replacing it with zero), we do not have a term u_{-2}^k in the system. For $i = 1$, however, we have

$$\frac{\mathrm{d}}{\mathrm{d}t} v_1^k = -\frac{EI}{\rho} \left[\frac{u_{-1}^k - 4u_0^k + 6u_1^k - 4u_2^k + u_3^k}{\Delta x^4} \right] + \frac{1}{\rho} p_1^k \tag{8.9}$$

where $p_i^k = p(x_i, t_k)$. The Neumann condition $\frac{\partial u}{\partial x}(0,t) = 0$ for the clamped beam can be discretised at $t = t_k$ using central differences: $\frac{u_1^k - u_{-1}^k}{2\Delta x}$, hence $u_{-1}^k = u_1^k$. Therefore, we can eliminate u_{-1}^k at $i = 1$:

$$\frac{\mathrm{d}}{\mathrm{d}t} v_1^k = -\frac{EI}{\rho} \left[\frac{-4u_0^k + 7u_1^k - 4u_2^k + u_3^k}{\Delta x^4} \right] + \frac{1}{\rho} p_1^k \tag{8.10}$$

Similarly, the boundary condition $\frac{\partial^2 u}{\partial x^2}(0,t) = 0$ for the simply supported can be discretised at $t = t_k$ using $\frac{u_1^k - 2u_0^k + u_{-1}^k}{\Delta x} = 0$, hence $u_{-1}^k = -u_1^k + 2u_0^k$. This results in the equation

$$\frac{\mathrm{d}}{\mathrm{d}t} v_1^k = -\frac{EI}{\rho} \left[\frac{-2u_0^k + 5u_1^k - 4u_2^k + u_3^k}{\Delta x^4} \right] + \frac{1}{\rho} p_1^k \tag{8.11}$$

After discretising, we arrive at the system of $n + 1$ equations

$$\begin{aligned} \frac{\mathrm{d}\mathbf{u}}{\mathrm{d}t} &= \mathbf{v} \\ \frac{\mathrm{d}\mathbf{v}}{\mathrm{d}t} &= A\mathbf{u} + \mathbf{f} \end{aligned} \tag{8.12}$$

Next, an appropriate time integration scheme should be used, such as the ones described in Chapter 5, in order to find an approximation of the solution.

8.3.3 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

$$\frac{\partial^2}{\partial t^2}u_j^k = \frac{u_j^{k+1} - 2u_j^k + u_j^{k-1}}{\Delta t^2} + \mathcal{O}(\Delta t^2) \tag{8.13}$$

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

$$u_j^{k+1} = \Delta t^2 \frac{EI}{\rho}\left[\frac{-u_{j-2}^k + 4u_{j-1}^k - 6u_j^k + 4u_{j+1}^k - u_{j+2}^k}{\Delta x^4}\right] + 2u_j^k - u_j^{k-1} \tag{8.14}$$

Note that equation (8.14) 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 [16]. 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 (8.14) and expanding the exponent, we find

$$g^{k+1}\mathrm{e}^{ij\xi} = \frac{EI\Delta t^2}{\rho\Delta x^4} g^k \mathrm{e}^{ij\xi}\left[-\mathrm{e}^{-2i\xi} + 4\mathrm{e}^{-i\xi} - 6 + 4\mathrm{e}^{i\xi} - \mathrm{e}^{2i\xi}\right] + 2g^k\mathrm{e}^{ij\xi} - g^{k-1}\mathrm{e}^{ij\xi} \tag{8.15}$$

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

$$g^2 = \frac{EI\Delta t^2}{\rho\Delta x^4} g\left[-\mathrm{e}^{-2i\xi} + 4\mathrm{e}^{-i\xi} - 6 + 4\mathrm{e}^{i\xi} - \mathrm{e}^{2i\xi}\right] + 2g - 1 \tag{8.16}$$

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

$$g^2 + \left(-2 - \frac{EI\Delta t^2}{\rho\Delta x^4}\left[-2\cos(2\xi) + 8\cos(\xi) - 6\right]\right) g + 1 = 0 \tag{8.17}$$

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

$$g_\pm = \frac{2 - 2y \pm \sqrt{(-2+2y)^2 - 4}}{2} = 1 - y \pm \sqrt{y^2 - 2y} \tag{8.18}$$

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

$$|g_\pm| = |1 - y \pm i\sqrt{2y - y^2}| = \sqrt{(1-y)^2 + |2y - y^2|} = \sqrt{1 - 2y + y^2 + 2y - y^2} = 1 \quad (8.19)$$

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

$$y = \frac{EI\Delta t^2}{\rho \Delta x^4} \leq \frac{2}{\cos(2\xi) - 4\cos(\xi) + 3} = \frac{1}{4} \quad (8.20)$$

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 (8.8) 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.

8.3.4 Implicit methods

As we have seen, explicit time integration methods can result in a strong CFL condition. To get rid of this condition, it might be a good idea to apply an implicit scheme 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 problems 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 then 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 implicit Backward Euler method, we can compute the solution analytically if we suppose that $\mathbf{f}^{k+1}$ is known beforehand. A single Backward Euler step is given by

$$\begin{aligned} \mathbf{u}^{k+1} &= \mathbf{u}^k + \Delta t \mathbf{v}^{k+1} \\ \mathbf{v}^{k+1} &= \mathbf{v}^k + \Delta t[\mathbf{A}^{k+1} + \mathbf{f}^{k+1}] \end{aligned} \quad (8.21)$$

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

$$\mathbf{z}^{k+1} = \mathbf{z}^k + \Delta t B \mathbf{z}^{k+1} + \Delta t \mathbf{g}^{k+1} \quad (8.22)$$

where

$$B = \begin{pmatrix} 0 & I \\ A & 0 \end{pmatrix} \qquad \text{and } \mathbf{g}^{k+1} = \begin{pmatrix} \mathbf{0} \\ \mathbf{f}^{k+1} \end{pmatrix} \quad (8.23)$$

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

$$\mathbf{z}^{k+1} = (I - \Delta t B)^{-1}(\mathbf{z}^k + \Delta t \mathbf{g}^{k+1}) \tag{8.24}$$

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.

8.4 Modal analysis

In many situations, we are interested in the eigenfrequencies of the solutions of equation (8.4). As a bridge deforms, multiple waves of different wave lengths appear. These waves are called mode shapes. As we will show shortly, these mode shapes only depend on the stiffness and the mass of the beam; they are independent on the pressure function p. Engineers can make sure that the frequency of the applied periodic force does not coincide with a modal frequency in order to prevent resonance.

In this section, we will consider a different approach to solve the beam equation. This approach makes use the mode shapes of the beam. First, we will discuss how the mode shapes and eigenfrequencies can be computed. Then, we will show how the response of each mode shape as the result of a pressure being exerted on the beam can be derived. Finally, the total response of the bridge is approximated by superposing the responses of each mode.

8.4.1 Mode shapes

We are interested in the frequencies that occur in the solutions of equation (8.4). The free vibrations of the beam 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 $v(x,t) = w(x)\mathrm{e}^{i\lambda t}$. 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

$$EIw^{(4)}(x)\mathrm{e}^{i\lambda t} = \rho\lambda^2 w(x)\mathrm{e}^{i\lambda t} \tag{8.25}$$

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

$$w^{(4)}(x) = \frac{\lambda^2\rho}{EI}w(x) \tag{8.26}$$

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 should satisfy $\beta^4 = \frac{\lambda^2\rho}{EI}$, and therefore

$$\beta = \left(\frac{\lambda^2\rho}{EI}\right)^{1/4} \tag{8.27}$$

We write the solution w as

$$w(x) = c_1\cos(\beta x) + c_2\sin(\beta x) + c_3\cosh(\beta x) + c_4\sinh(\beta x) \tag{8.28}$$

Note that if $\beta = 0$, it follows immediately that w is a constant. It must therefore be identically zero due to the boundary conditions. Since we are only looking for non-trivial solutions, we can assume that $\beta \neq 0$.

8.4.2 Boundary conditions

For the clamped beam, we must have $v(0) = v(L) = \frac{\partial v}{\partial x}(0) = \frac{\partial v}{\partial x}(L)$. w satisfies the boundary equation if and only if

$$\begin{aligned} 0 &= v(0) = c_1 + c_3 \\ 0 &= \frac{\partial v}{\partial x}(0) = \beta(c_2 + c_4) \\ 0 &= v(L) = c_1\cos(\beta L) + c_2\sin(\beta L) + c_3\cosh(\beta L) + c_4\sinh(\beta L) \\ 0 &= \frac{\partial v}{\partial x}(L) = \beta(-c_1\sin(\beta L) + c_2\cos(\beta L) + c_3\sinh(\beta L) + c_4\cosh(\beta L)) \end{aligned} \tag{8.29}$$

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

$$\begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ \cos(\beta L) & \sin(\beta L) & \cosh(\beta L) & \sinh(\beta L) \\ -\sin(\beta L) & \cos(\beta L) & \sinh(\beta L) & \cosh(\beta L) \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \tag{8.30}$$

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

$$\cos(\beta L)\cosh(\beta L) = 1 \tag{8.31}$$

Equation (8.31) has infinitely many solutions. Since $\cosh(\beta L) > 1$, $\cos(\beta L) > 0$ must hold. Furthermore, since $\cosh(\beta L)$ tends to infinity as $\beta \to \infty$, $\cos(\beta L)$ must be near zero for large β. Therefore, the n-th solution β_n tends to $\frac{1}{L}(\frac{1}{2} + n)\pi$ for $n \geq 1$ as n tends to infinity. The solutions of (8.31) can be approximated by using a numerical method like Newton-Raphson.

The actual frequencies λ_n of the beam are then easily derived from equation (8.27):

$$\lambda_n = \beta_n^2\sqrt{\frac{EI}{\rho}} \tag{8.32}$$

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 (8.31) is true, so that this vector is unique up to a scale factor.

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

$$w_n(x) = \cosh(\beta_n x) - \cos(\beta_n x) + k_n\sin(\beta_n x) - k_n\sinh(\beta_n x) \tag{8.33}$$

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

$$k_n = \frac{\sin(\beta_n L)\sinh(\beta_n L)}{\cos(\beta_n L)\sinh(\beta_n L) - \sin(\beta_n L)\cosh(\beta_n L)} \tag{8.34}$$

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 (8.31), we find the following first 5 values of β_n and λ_n as in Table 8.1. 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 8.1: The first 5 frequencies of the clamped beam using $E = I = \rho = L = 1$.

As one can see, the approximation $\frac{1}{L}(\frac{1}{2}+n)\pi$ converges very quickly to the value of β_n. Note that $\beta = 0$ also satisfies (8.31), but this corresponds to a zero modal function. Furthermore, since $\lambda_n = \beta_n^2\sqrt{\frac{EI}{\rho}}$, we can ignore negative values of n too.

In Figure 8.2 the first five mode shapes are shown for the clamped beam.

From the boundary conditions for the simply supported beam, we find

$$\begin{aligned}
0 &= v(0) = c_1 + c_3 \\
0 &= \frac{\partial^2 v}{\partial x^2}(0) = \beta^2(-c_1 + c_3) \\
0 &= v(L) = c_1\cos(\beta L) + c_2\sin(\beta L) + c_3\cosh(\beta L) + c_4\sinh(\beta L) \\
0 &= \frac{\partial^2 v}{\partial x^2}(L) = \beta^2(-c_1\cos(\beta L) - c_2\sin(\beta L) + c_3\sinh(\beta L) + c_4\cosh(\beta L))
\end{aligned} \tag{8.35}$$

It follows directly from the first two equations that $c_1 = c_3 = 0$, since β is assumed to be non-zero. The third and fourth equation yield $0 = c_4(\sinh(\beta L) + \cosh(\beta L)) = c_4 e^{\beta L}$, hence $c_4 = 0$. Therefore, $w(x) = c_2\sin(\beta x)$. For non-trivial solutions we must have $c_2 \neq 0$, and therefore $\sin(\beta L) = 0$, i.e. βL is a multiple of π:

$$\beta_n = \frac{n\pi}{L} \tag{8.36}$$

8.4.3 Orthogonality of the mode shapes

An important property of the mode shapes for both the clamped beam as well as the simply supported beam, is that they are orthogonal. To prove this, we will first proof the following important Lemma.

Lemma 8.1. For two mode shapes w_i and w_j, either for the case with the clamped beam or the simply supported beam, the following equation must hold:

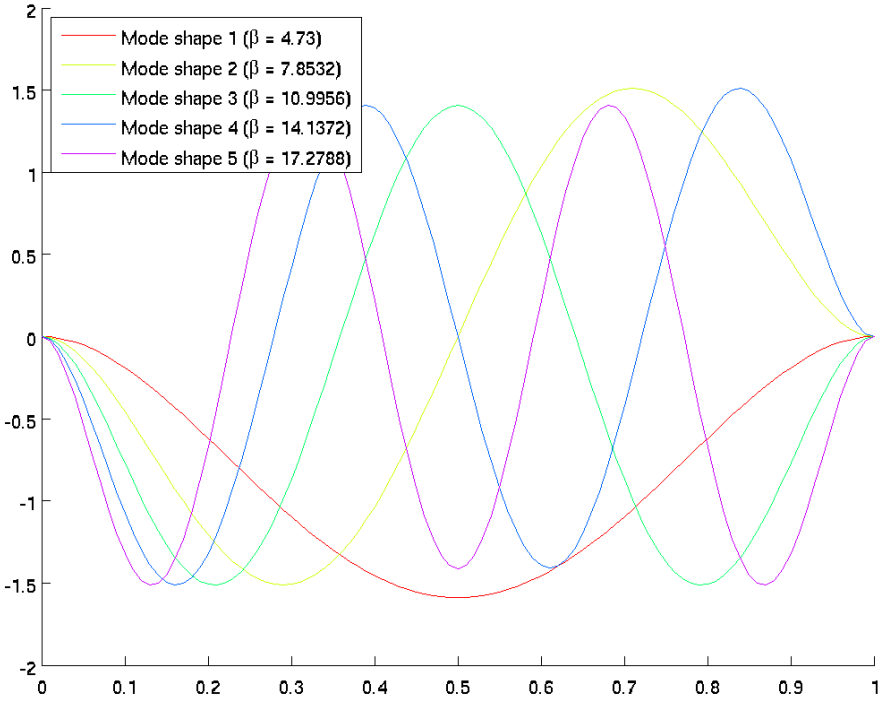


Figure 8.2: The first five mode shapes of the clamped beam.

$$\int_0^L EI\frac{\partial^4 w_i}{\partial x^4}w_j \mathrm{d}x = \int_0^L EI\frac{\partial^4 w_j}{\partial x^4}w_i \mathrm{d}x \tag{8.37}$$

Proof: We integrate by parts twice. We have

$$\begin{aligned}\int_0^L EI\frac{\partial^4 w_i}{\partial x^4}w_j \mathrm{d}x &= \left[EI\frac{\partial^3 w_i}{\partial x^3}w_j\right]_0^L - \int_0^L EI\frac{\partial^3 w_i}{\partial x^3}\frac{\partial w_j}{\partial x}\mathrm{d}x \\ &= \left[EI\frac{\partial^3 w_i}{\partial x^3}w_j\right]_0^L - \left[EI\frac{\partial^2 w_i}{\partial x^2}\frac{\partial w_j}{\partial x}\right]_0^L + \int_0^L EI\frac{\partial^2 w_i}{\partial x^2}\frac{\partial^2 w_j}{\partial x^2}\mathrm{d}x\end{aligned} \tag{8.38}$$

The first term is zero since $w_j(0) = w_j(L) = 0$. For the clamped beam, the second term is also zero because $\frac{\partial w_j}{\partial x}(0) = \frac{\partial w_j}{\partial x}(L) = 0$. Additionally, for the simply supported beam, this term is zero as well since $\frac{\partial^2 w_i}{\partial x^2}(L) = 0$.

It now follows directly that

$$\begin{aligned}\int_0^L EI\frac{\partial^4 w_i}{\partial x^4}w_j &= \int_0^L EI\frac{\partial^2 w_i}{\partial x^2}\frac{\partial^2 w_j}{\partial x^2}\mathrm{d}x \\ &= \int_0^L EI\frac{\partial^2 w_j}{\partial x^2}\frac{\partial^2 w_i}{\partial x^2}\mathrm{d}x = \int_0^L EI\frac{\partial^4 w_j}{\partial x^4}w_i\mathrm{d}x\end{aligned} \tag{8.39}$$

which proves the Lemma. □

Using the Lemma, proving the orthogonality of the mode shapes is fairly straightforward:

Theorem 8.2. For both the clamped beam and the simply supported beam, the mode shapes are orthogonal.

Proof: Let w_i and w_j be mode shapes with corresponding eigenfrequencies λ_i and λ_j. Since both $v_i(x,t) = w_i(x)\mathrm{e}^{i\lambda_i t}$ and $v_j(x,t) = w_j(x)\mathrm{e}^{i\lambda_j t}$ satisfy equation (8.4), we have

$$EI\frac{\partial^4 w_i}{\partial x^4}(x)\mathrm{e}^{i\lambda_i t} = -\rho\frac{\partial^2 u}{\partial t^2}\left[w(x)\mathrm{e}^{i\lambda_i t}\right] = \rho\lambda_i^2 w(x)\mathrm{e}^{i\lambda_i x} \tag{8.40}$$

and therefore

$$EI\frac{\partial^4 w_i}{\partial x^4}(x) + \rho\lambda_i^2 w_i(x) = 0 \tag{8.41}$$

Next, we multiply by $w_j(x)$ and integrate from $x = 0$ to $x = L$:

$$\begin{aligned}0 &= \int_0^L \left(EI\frac{\partial^4 w_i}{\partial x^4}(x)w_j(x) + \rho\lambda_i^2 w_i(x)w_j(x)\right)\mathrm{d}x \\ &= \int_0^L \left(EI\frac{\partial^4 w_j}{\partial x^4}(x)w_i(x) + \rho\lambda_i^2 w_i(x)w_j(x)\right)\mathrm{d}x \\ &= \int_0^L \left(EI\frac{\partial^4 w_j}{\partial x^4}(x)w_i(x) + \rho\lambda_j^2 w_i(x)w_j(x)\right)\mathrm{d}x + \rho(\lambda_i^2 - \lambda_j^2)\int_0^L w_i(x)w_j(x)\mathrm{d}x \\ &= \rho(\lambda_i^2 - \lambda_j^2)\int_0^L w_i(x)w_j(x)\mathrm{d}x\end{aligned} \tag{8.42}$$

where we applied Lemma 8.1 for the first equation. If $\lambda_i \neq \lambda_j$, then $\rho(\lambda_i^2 - \lambda_j^2) \neq 0$, hence we can conclude that $\int_0^L w_i(x)w_j(x)\mathrm{d}x = 0$. If $\lambda_i = \lambda_j$, then $\int_0^L w_i(x)^2\mathrm{d}x > 0$, since we did not allow the zero function to be a mode shape. Therefore, the mode shapes are orthogonal. □

Additionally, since orthogonality holds, we can safely assume that the mode shapes are orthonormal by dividing each mode shape by its norm.

8.4.4 Modal analysis using the discretisation

Another approach is to use the finite difference discretisation matrix A to compute the eigenfrequencies and to construct the mode shapes. The exact mode shapes satisfy $w_i^{(4)}(x) = \beta_i^4 w(x)$. For a discretised mode shape, we have

$$A\mathbf{w}(x) \approx \frac{EI}{\rho}\mathbf{w}_i^{(4)}(x) = \frac{EI}{\rho}\beta_i^4\mathbf{w}(x) = \lambda^2\mathbf{w}(x) \tag{8.43}$$

Hence, it seems that the eigenvectors of the matrix A satisfy a similar equation as the mode shapes. The corresponding eigenvalues seem to be equal to the square of the eigenfrequencies. Therefore, it is worth checking whether these eigenvectors and eigenvalues are approximations of the mode shapes and eigenfrequencies.

λ	$\sqrt{\mu}$, $n_x = 20$	$\sqrt{\mu}$, $n_x = 200$
28.7004	28.3771	28.6974
79.1137	77.1775	79.0956
155.0946	148.6791	155.0336
256.3792	240.4848	256.2257
382.9863	350.0241	382.6624

Table 8.2: Comparison between the analytical eigenfrequencies λ and $\sqrt{\mu}$, where μ represents the eigenvalues of the discretisation matrix A.

As an example, consider a bridge with clamped boundaries using the elastic properties as given in Table 8.3. The first three eigenvectors and their corresponding eigenfrequencies are computed. This is done both by using 20 discretisation points as well as 200. Note that the first two eigenvalues of A, both with value 1, are ignored. These eigenvalues are results of the implementation of the boundary conditions.

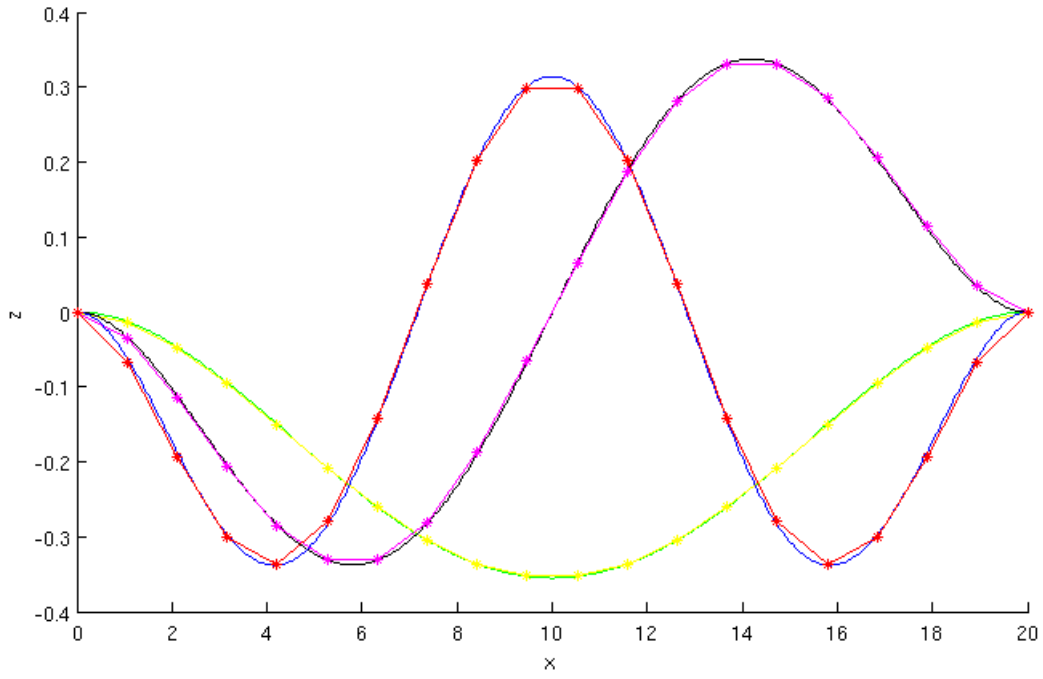


Figure 8.3: Comparison between the first three mode shapes and the eigenvectors computed for $n_x = 20$

As one can see, the eigenfrequencies correspond well with the square root of the eigenvalues

of A. The difference between the two is clearly much smaller if the discretisation matrix is larger. However, the approximation becomes worse for larger eigenfrequencies. An important property to mention is that the discretisation matrix A only has n eigenvalues (or $n-2$ if the two 1-eigenvalues are ignored), yet there are an infinite amount of eigenfrequencies. Hence the larger eigenvalues and their corresponding eigenvectors might be the combination of multiple eigenfrequencies and mode shapes.

Similarly, we compare the eigenvectors with the mode shapes. Note that if $\mathbf{w}$ is an eigenvectors, any (non-zero) multiple is an eigenvector too. We have assumed that the mode shapes are normalised, hence we should multiply each eigenvector by a constant so that its interpolated function has norm 1, i.e. we set

$$\mathbf{w} := \mathbf{w} / \sqrt{\Delta x \sum_{i=1}^{n} \mathbf{w}_i^2} \tag{8.44}$$

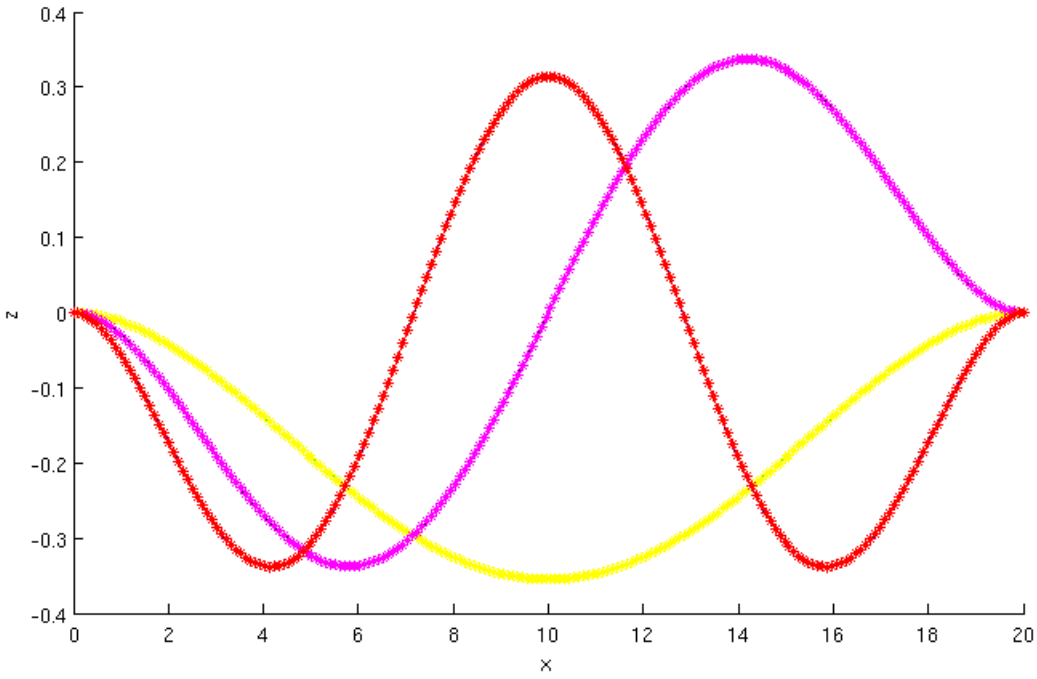


Figure 8.4: Comparison between the first three mode shapes and the eigenvectors computed for $n_x = 200$. The eigenvectors seem to approximate the mode shapes perfectly.

See Figure 8.3 and 8.4. Here, the first 3 (analytical) mode shapes as well as the normalised eigenvectors corresponding to the smallest 3 eigenvalues (excluding the two eigenvalues with value 1) are shown. The eigenvectors clearly approximate the mode shapes fairly well. Similarly to the result for the eigenvalues, the approximation is better for smaller eigenvalues and increasing the number of mesh points n_x increases the accuracy of the solution.

This approach is useful if the mode shapes and/or the corresponding eigenfrequencies are not known beforehand. For example, if the beam is not homogeneous (i.e. E,I, and/or ρ are not constant), then the mode shapes do not satisfy (8.26). In this situation, the derived

differential equation for w might be impossible to solve analytically. To get an approximation of the mode shapes and eigenfrequencies, one could therefore compute the eigenvalues and eigenvectors of the discretisation matrix instead.

Note that the eigenvectors only correspond to amplitudes for certain x. To compute $w_i(x)$ for some general $x \in [0, L]$, one may need to interpolate with respect to the discrete set $\{x_i | 0 \leq i \leq n\}$. Next, the coefficients $c_i(t)$ for $i = 1, \ldots, m$ can be computed, and the modal approximation u_m can be constructed. The eigenfrequencies λ can be computed by taking the square root of the eigenvalues μ of the matrix.

8.5 The stationary case

8.5.1 The analytic solution

Assume that the pressure is independent of the time. we will write the solution of equation (8.4) as a superposition of the mode shapes. First, we will only look at the stationary situation, i.e. the solution of the stationary beam equation

$$EI \frac{\mathrm{d}^4 u}{\mathrm{d}x^4} = p(x) \tag{8.45}$$

Using superposition the solution u can be written as

$$u(x) = \sum_{i=1}^{\infty} c_i w_i(x) \tag{8.46}$$

where w_i is the i^{th} mode shape and $c_i \in \mathbb{R}$ for all $i \geq 1$. In order to compute the coefficients c_i, we first substitute expression (8.46) into differential equation (8.45):

$$\begin{aligned} p(x) &= EI \frac{\partial}{\partial x^4} \left[\sum_{i=1}^{\infty} c_i w_i(x) \right] = EI \sum_{i=1}^{\infty} c_i w_i^{(4)}(x) \\ &= EI \sum_{i=1}^{\infty} c_i \beta_i^4 w_i(x) = \rho \sum_{i=1}^{\infty} c_i \lambda_i^2 w_i(x) \end{aligned} \tag{8.47}$$

where equation (8.26) is applied.

Next, we multiply (8.47) by a mode shape $w_j(x)$ and then integrate from $x = 0$ to $x = L$. By the orthonormality of the mode shapes, we find

$$\begin{aligned} \int_0^L p(x) w_j(x) \mathrm{d}x &= \rho \int_0^L \sum_{i=1}^{\infty} c_i \lambda_i^2 w_i(x) w_j(x) \mathrm{d}x = \rho \sum_{i=1}^{\infty} c_i \lambda_i^2 \int_0^L w_i(x) w_j(x) \mathrm{d}x \\ &= \rho c_j \lambda_j^2 \end{aligned} \tag{8.48}$$

And therefore, the coefficients c_j are equal to

$$c_j = \frac{1}{\rho\lambda_j^2}\int_0^L p(x)w_j(x)\mathrm{d}x \tag{8.49}$$

8.5.2 Numerical approximation using mode shapes

The goal is to compute the coefficients c_i and then approximate u by using a finite (say m) number of modes:

$$u(x) \approx u_m(x) = \sum_{i=1}^{m} c_i w_i(x) \tag{8.50}$$

The coefficients c_i are given by (8.49). In most practical situations, the integral cannot be computed analytically. Hence, a numerical integrator should be used.

Computing the Riemann sum is the easiest way to approximate the integral. If a better approximation is required, the Trapezoidal rule or a higher order Newton-Cotes formula can be used.

It is worth noting that in many cases, the pressure $p(x)$ is only non-zero in a relatively small region. For example, the contact area of a train wheel on a bridge is very small, usually a few millimeters at most, compared with the length of the complete bridge, which spans 10 meter easily. Therefore, computing this integral accurately requires far less grid points than discretisating the entire bridge using the Finite Difference or the Finite Element method.

In Figure 8.5 the first four mode shapes for a simply supported beam is shown. The black line represents the support of the pressure function. As an example, this can represent the pressure of a wheel on a bridge located at $x = 1/2$. The mode shapes represented by the dashed lines are odd function with respect to $x = 1/2$. Therefore the corresponding coefficients c_i (for i even) are approximately equal to zero. If p is assumed to be even around $x = 1/2$, then $p(x)w_i(x)$ is odd, hence $c_i = 0$ for i even.

8.5.3 Error analysis

Theorem 8.3. For both the clamped beam and the simply supported beam, let $p \in \mathcal{L}_2(0, L)$ and u the solution of the corresponding beam equation, and u_m the approximation using m modes. Then we have the following error estimate:

$$\|u - u_m\|_2 \leq \frac{L^4\|p\|_2}{3\pi^4 EIm^3} \tag{8.51}$$

Proof: By the triangle inequality,

$$\begin{aligned}\|u - u_m\|_2 &= \|\sum_{i=1}^{\infty} c_i w_i - \sum_{i=1}^{m} c_i w_i\|_2 = \|\sum_{i=m+1}^{\infty} \frac{1}{\rho\lambda_i^2}\left[\int_0^L p(x)w_i(x)\mathrm{d}x\right] w_i\|_2 \\ &\leq \sum_{i=m+1}^{\infty} \frac{1}{\rho\lambda_i^2}\left|\int_0^L p(x)w_i(x)\mathrm{d}x\right| \|w_i\|_2\end{aligned} \tag{8.52}$$

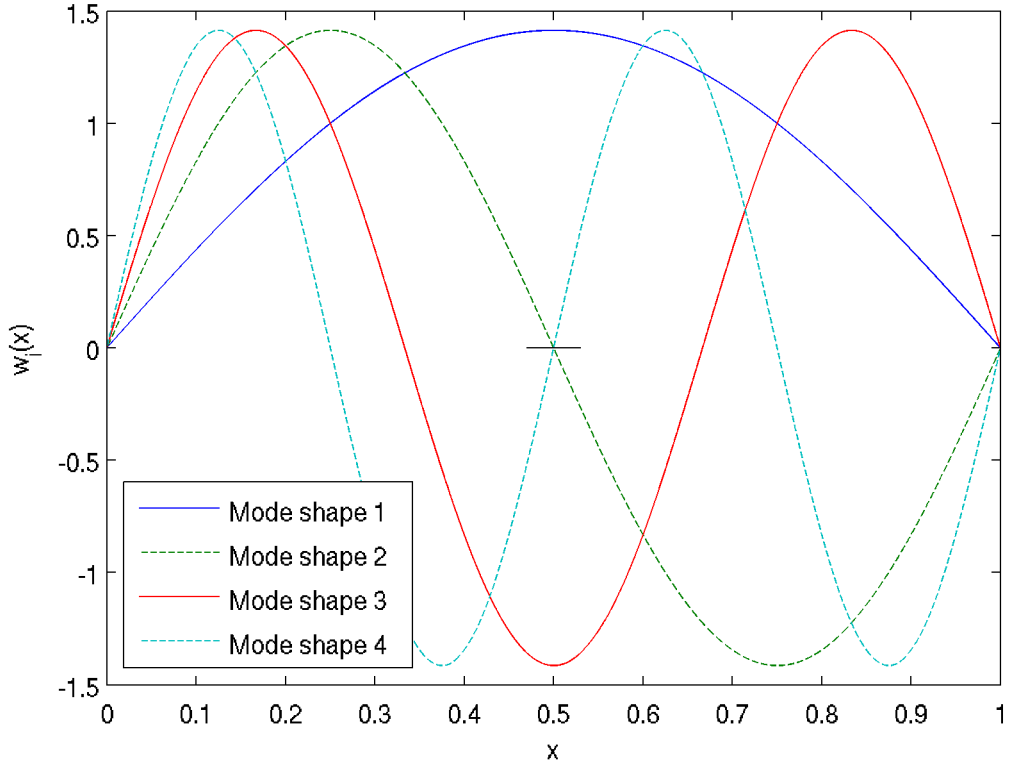


Figure 8.5: The first four modes for a simply supported beam.

By the Cauchy-Schwarz inequality, since $\|w_i\|_2 = 1$, it follows that

$$\begin{aligned}\|u - u_m\|_2 &\leq \sum_{j=m+1}^{\infty} \frac{1}{\rho\lambda_i^2} \left| \int_0^L p(x) w_i(x) \mathrm{d}x \right| \leq \sum_{j=m+1}^{\infty} \frac{1}{\rho\lambda_i^2} \|p\|_2 \|w_i\|_2 \\ &= \sum_{j=m+1}^{\infty} \frac{1}{\rho\lambda_i^2} \|p\|_2 = \|p\|_2 \sum_{j=m+1}^{\infty} \frac{1}{\beta_j^4 EI} \\ &\leq \|p\|_2 \sum_{j=m+1}^{\infty} \frac{1}{\left(\frac{j\pi}{L}\right)^4 EI} = \frac{L^4 \|p\|_2}{\pi^4 EI} \sum_{j=m+1}^{\infty} \frac{1}{j^4}\end{aligned} \tag{8.53}$$

Since the function $x \mapsto \frac{1}{x^4}$ is a decreasing function for $x > 0$, we can bound the sum $\sum_{j=m+1}^{\infty} \frac{1}{j^4}$ by the integral $\int_m^{\infty} \frac{1}{x^4} \mathrm{d}x$. Hence

$$\begin{aligned}\|u - u_m\|_2 &\leq \frac{L^4 \|p\|_2}{\pi^4 EI} \int_m^{\infty} \frac{1}{x^4} \mathrm{d}x = \frac{L^4 \|p\|_2}{\pi^4 EI} \left[-\frac{1}{3x^3} \right]_m^{\infty} \\ &= \frac{L^4 \|p\|_2}{3\pi^4 EI m^3}\end{aligned} \tag{8.54}$$

which is exactly what was to be shown. □

Hence, given a function p and constants E and I, not only does the modal approach converge to the exact solution, the error is of order $\mathcal{O}(m^{-3})$. The error can therefore by decreased by a factor 1000 by simply taking 10 times as many modes in the approximation.

In most practical situations, we are interested in the reverse of this Theorem. The following Corollary tells us how many modes are needed to find an accurate result.

Corollary 8.4. Given $\varepsilon > 0$, the error for the modal approximation is $\|u - u_m\|_2 < \varepsilon$, if m satisfies

$$m > \left(\frac{\varepsilon L^4 \|p\|_2}{3\pi^4 EI} \right)^{1/3} \tag{8.55}$$

Proof: This is direct consequence of Theorem 8.3. □

8.5.4 Numerical validation

Consider the following (arbitrary) example. Suppose $L = E = I = 1$ and consider the beam to be simply supported at both $x = 0$ and $x = L$. Let p by given by

$$p(x) = \begin{cases} -1 & \text{for } x \in [.299, .301] \\ 2 & \text{for } x \in [.899, .901] \\ 0 & \text{otherwise} \end{cases} \tag{8.56}$$

The solution of the stationary beam equation is approximated by using 1,2, or 3 mode shapes. The result can be seen in Figure 8.6. The figure includes the result of the finite difference method, using a very small mesh size $\Delta x = 1/1000$.

The approximation using one mode shape is constructed using only a half sine function, and is therefore not accurate. However, as we can see, the shape of the finite difference solution is matched by using only 2 modes. Using only 7 mode shapes, the difference between the finite difference solution and the modal solution cannot be distinguished in a similar graph.

See Appendix A.4 for the Matlab code implementing both the discrete approach as well as the modal approach. Here, the finite difference solution using a very small mesh is computed as well as the modal solution using a variable number of mode shapes.

8.6 Forced vibrations

8.6.1 The differential equation

A similar approach can be taken to approximate the solution of the time-dependent differential equation (8.1). Now, the external force $p(x, t)$ is also allowed to be a function of the time. For the moving load problem, as an example, the external force moves as wave from one side to the other. This force can trigger vibrations of different frequencies.

Using superposition, we can write the solution of (8.4) as

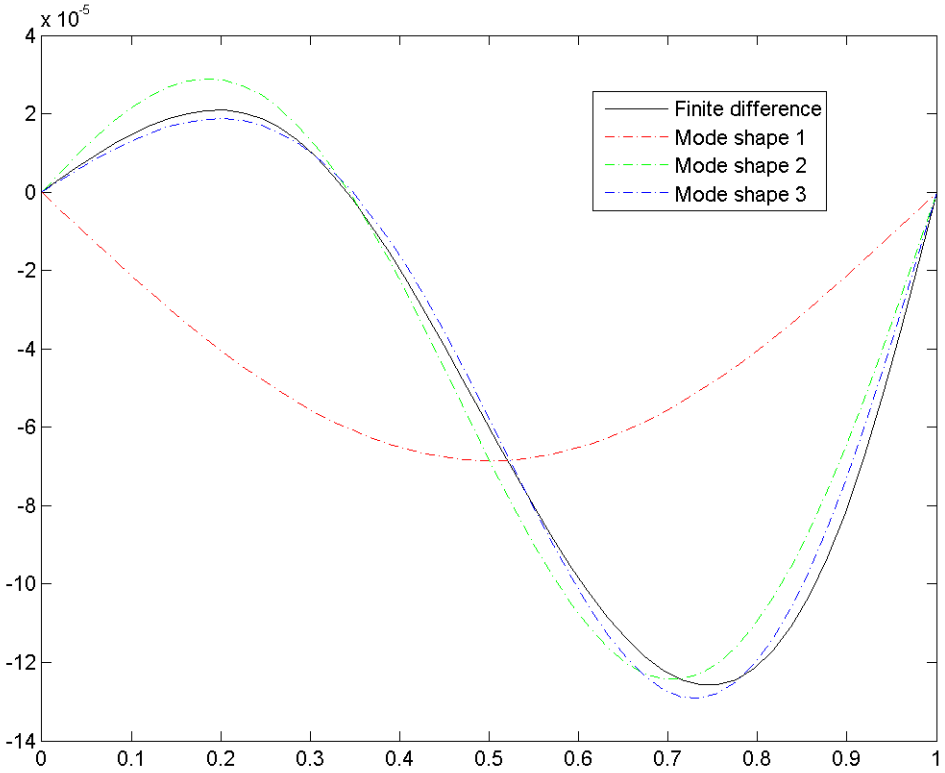


Figure 8.6: The approximation of the stationary beam equation using the first 3 mode shapes.

$$u(x) = \sum_{i=1}^{\infty} c_i(t) w_i(x) \tag{8.57}$$

The difference with the stationary case is that the coefficients $c_i(t)$ are not necessarily constant and can vary with respect to time. To compute the coefficients, we substitute (8.57) into (8.4):

$$\begin{aligned} p(x,t) &= EI \frac{\partial}{\partial x^4} \left[\sum_{i=1}^{\infty} c_i(t) w_i(x) \right] + \rho \frac{\partial}{\partial t^2} \left[\sum_{i=1}^{\infty} c_i(t) w_i(x) \right] \\ &= \sum_{i=1}^{\infty} \left[EI c_i(t) w_i^{(4)}(x) + \rho c_i''(t) w_i(x) \right] \\ &= \sum_{i=1}^{\infty} w_i(x) \left[EI c_i(t) \beta_i^4 + \rho c_i''(t) \right] \end{aligned} \tag{8.58}$$

Next, we multiply by a mode shape $w_j(x)$ and integrate both sides from $x = 0$ to $x = L$: Since the mode shapes are orthonormal, we have

$$\begin{aligned}\int_0^L p(x,t)w_j(x)\mathrm{d}x &= \int_0^L \sum_{i=1}^{\infty} w_i(x)w_j(x)\left[EIc_i(t)\beta_i^4 + \rho c_i''(t)\right]\mathrm{d}x \\ &= \sum_{i=1}^{\infty}\left[EIc_i(t)\beta_i^4 + \rho c_i''(t)\right]\int_0^L w_i(x)w_j(x)\mathrm{d}x \\ &= EIc_j(t)\beta_j^4 + \rho c_j''(t)\end{aligned} \tag{8.59}$$

The coefficients c_j therefore each satisfy an ordinary differential equation. It is important to note that the differential equations corresponding to different coefficients $c_i(t)$ and $c_j(t)$ $(i \neq j)$ are independent of each other.

8.6.2 Numerical approximation using mode shapes

Similarly to the stationary case, the solution of (8.1) can be approximated by using m modes:

$$u(x,t) \approx u_m(x,t) = \sum_{i=1}^{m} c_i(t)w_i(x) \tag{8.60}$$

To compute this approximation, we need to solve m independent differential equations for $c_i(t)$, $1 \leq i \leq m$. For each differential equation, we also need initial conditions for $c_i(0)$ and $c_i'(0)$. Differential equation (8.59) is of second order, hence both $c_i(0)$ as well as $c_i'(0)$ should be given.

In most situations, both $u(x,0) = u_0(x)$ and $v(x,0) = v_0(x)$ are known. Using the orthonormality of the mode shapes, the initial conditions for c_i can be extracted:

$$\begin{aligned}c_j(0) = c_j(0)\sum_{i=1}^{\infty}\int_0^L w_i(x)w_j(x)\mathrm{d}x &= \sum_{i=1}^{\infty} c_j(0)\int_0^L w_i(x)w_j(x)\mathrm{d}x \\ = \int_0^L \left[\sum_{i=1}^{\infty} c_i(0)w_i(x)\right]w_j(x)\mathrm{d}x &= \int_0^L u(x,0)w_j(x)\mathrm{d}x \\ = \int_0^L u_0(x)w_j(x)\mathrm{d}x\end{aligned} \tag{8.61}$$

and similarly for $c_j'(0)$:

$$c_j'(0) = \int_0^L \left[\sum_{i=1}^{\infty} c_i'(0)w_i(x)\right]w_j(x) = \int_0^L v_o(x)w_j(x)\mathrm{d}x \tag{8.62}$$

In most problems, both the displacement as well as the velocity is zero at $t = 0$. It follows from (8.61) and (8.62) that $c_j(t) = c_j'(t) \equiv 0$ for all $j \geq 1$.

Note that (8.59) can be rewritten as

$$c_j''(t) = \frac{1}{\rho}\int_0^L p(x,t)w_j(x)\mathrm{d}x - \frac{EI}{\rho}\beta_j^4 c_j(t) \qquad \text{for all } 1 \leq j \leq m \tag{8.63}$$

Note that for each separate mode shape w_i, differential equation (8.63) is independent on w_j for all $j \leq i$. Therefore, each differential equation corresponding to a different mode shape can be solved separately. These ordinary differential equation (8.63) can be solved using an integration scheme. This can be solved in a similar way as we have described before, see Section 8.3.4 for the implementation of an implicit integration scheme.

The right hand-side of (8.63) involves the computation of an integral. This integral should be approximated at each iteration using an integrator such as the trapezoidal rule or a similar Newton-Cotes formula.

8.7 A simple moving load problem

Consider the following (very simplistic) problem with a train 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. The train is modelled as a single point that exerts a constant (moving) point force on the track.

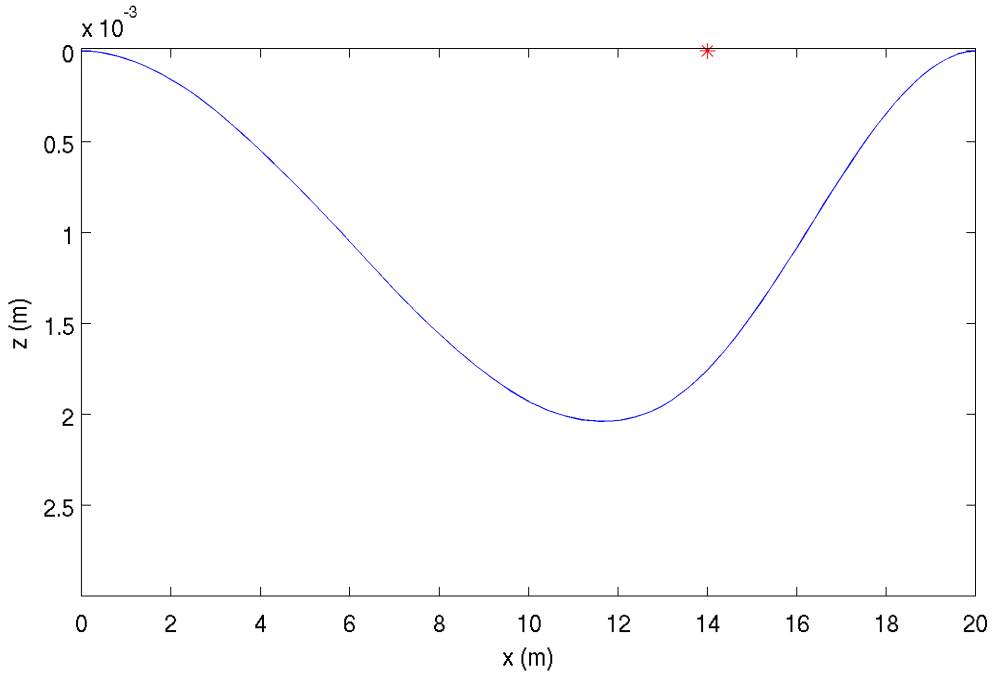


Figure 8.7: The displacement of the beam at time $t \approx 4.67$. The red star represents the load of the train.

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 8.3.

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

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 8.3: The properties of the bridge and train.

$$p(x,t) = mg\delta(x - vt) \qquad 0 \leq x \leq L \tag{8.64}$$

where δ is the Dirac delta function. The resulting force is therefore

$$F(t) = \int_0^L p(x,t)\mathrm{d}x = mg \int_0^L \delta(x - vt)\mathrm{d}x = mg \tag{8.65}$$

which is indeed equal to the gravitational force of the train.

8.7.1 The discrete problem

Both the finite difference method as well as the modal approach will be analysed and used to solved this problem. For the finite difference approach, (8.4) can be rewritten as two linear ordinary differential equations by defining $w = \frac{\partial u}{\partial t}$ and using finite differences to approximate the derivatives. Next, a time integration scheme such as the ones we have described in Chapter 5 can be used to integrate with respect to time. Alternatively, the partial differential equation can also be discretised completely, i.e. including the time derivative, and then solving the equation for the displacement and velocity components for the next iteration.

The algorithm to solve this problem (with the properties as shown in Table 8.3) has been implemented in Matlab. We use a uniform mesh in the spatial direction with mesh size $\Delta x = \frac{L}{100}$. The time integration is done using Verlet integration. According to (8.20), the following CFL condition should hold:

$$\Delta t \leq \frac{(\Delta x)^2}{2}\sqrt{\frac{\rho}{EI}} \approx 3.8977 \cdot 10^{-5} \tag{8.66}$$

In Figure 8.7 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 8.8.

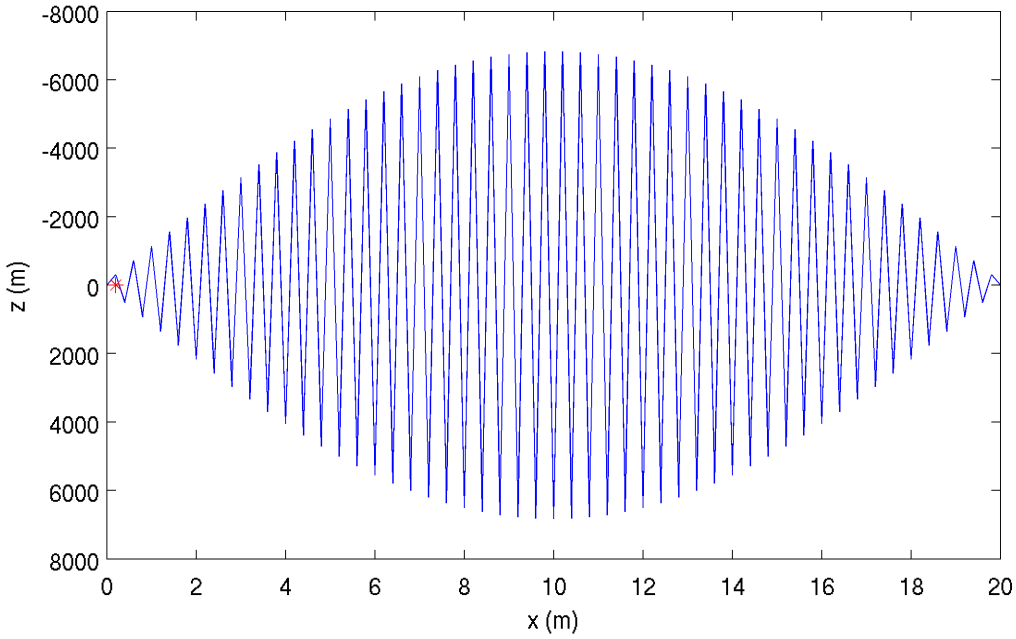


Figure 8.8: The displacement of the beam at time $t \approx 0.06$, using a CFL number slightly larger than $\frac{1}{4}$. The red star represents the position of the wheel.

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

The CFL condition (8.20) was computed for the fully discretised model, i.e. the Verlet method. The Forward Euler 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:

$$\Delta t \leq 0.188 \frac{\rho}{EI} \Delta x^4 = \mathcal{O}(\Delta x^4) \tag{8.67}$$

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 Forward Euler 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.

8.7.2 The modal approach

Similarly, the modal approach (using $m = 10$ modes) is applied to solve this moving load problem. There is no CFL condition for the modal approach regardless of the choice of

(explicit) time integration schemes. This is because each modal coefficient is prescribed by an ordinary differential equation; this is solved without the need of a mesh width Δx. Additionally, the solution u_m only consists of waves with bounded wave lengths and therefore cannot behave as in (8.8).

n	β_n	$\frac{1}{L}(\frac{1}{2}+n)\pi$	λ_n	Frequency $(\lambda_n/(2\pi))$
1	0.23650	0.23561	28.700	4.567Hz
2	0.39266	0.39269	79.113	12.59Hz
3	0.54978	0.54977	155.09	24.68Hz
4	0.70685	0.70685	256.37	40.80Hz
5	0.86393	0.86393	382.98	60.95Hz

Table 8.4: The first 5 eigenfrequencies of the bridge.

Table 8.4 shows the corresponding values of λ_n and β_n for the free vibration modes, as well as the corresponding frequencies.

As an example, Verlet integration can be applied to solve (8.63). For each $1 \leq i \leq m$, we set

$$\begin{aligned} c_i^1 &= c_i^0 + \Delta t \dot{c}_i^0 + \frac{1}{2}(\Delta t)^2 \left[\frac{1}{\rho}\int_0^L p(x,t_0)w_i(x)\mathrm{d}x - \frac{EI}{\rho}\beta_i^4 c_i^0\right] \\ c_i^{k+1} &= 2c_i^k - c_i^{k-1} + (\Delta t)^2 \left[\frac{1}{\rho}\int_0^L p(x,t^k)w_i(x)\mathrm{d}x - \frac{EI}{\rho}\beta_i^4 c_i^0\right] \qquad \text{for } k \geq 1 \end{aligned} \tag{8.68}$$

which is by definition of p equal to

$$\begin{aligned} c_i^1 &= c_i^0 + \Delta t \dot{c}_i^0 + \frac{1}{2}(\Delta t)^2 \left[\frac{mg}{\rho}w_i(vt_0) - \frac{EI}{\rho}\beta_i^4 c_i^0\right] \\ c_i^{k+1} &= 2c_i^k - c_i^{k-1} + (\Delta t)^2 \left[\frac{mg}{\rho}w_i(vt^k) - \frac{EI}{\rho}\beta_i^4 c_i^0\right] \qquad \text{for } k \geq 1 \end{aligned} \tag{8.69}$$

The resulting solution $u^m(x,t^k) = \sum_{i=1}^m c_i^k w_i(x)$ behaves similar to the static case. The coefficients c_i^k tend to 0 very quickly as i tends to infinity. This results in a similar situation as shown in Figure 8.6.

8.7.3 Resonance

An advantage of applying the modal approach is that because resonance can easily be analysed. If the exerted pressure p is periodic and its frequency corresponds to any of the eigenfrequencies, then the corresponding modal coefficient (and hence the total solution) will amplify.

Suppose the pressure as in (8.64) is $L-$periodic, i.e.

$$p(x,t) = -mg\delta(\mathrm{mod}(x-vt, L)) \tag{8.70}$$

This can represent the total exerted force of a train caused by the multiple wheels. We are interested in the influence of the velocity v of the train.

As computed in Table 8.4, the lowest eigenfrequency of the beam is equal to 4.567Hz. If the speed v of the train is equal to $4.567L \approx 91.356$ m/s, then the first modal response integral $\int_0^L p(x,t)w_1(x,t)\mathrm{d}x$ is a periodic function. By definition, its frequency corresponds to the lowest eigenfrequency of the beam. To see whether resonance can indeed occur at this speed, we can compute the norm $\|u(t)\|_\infty$. We have done so for the 'critical' speed $v = 91.356$m/s, as well as a slightly different speed $v = 100$m/s. See Figure 8.9.

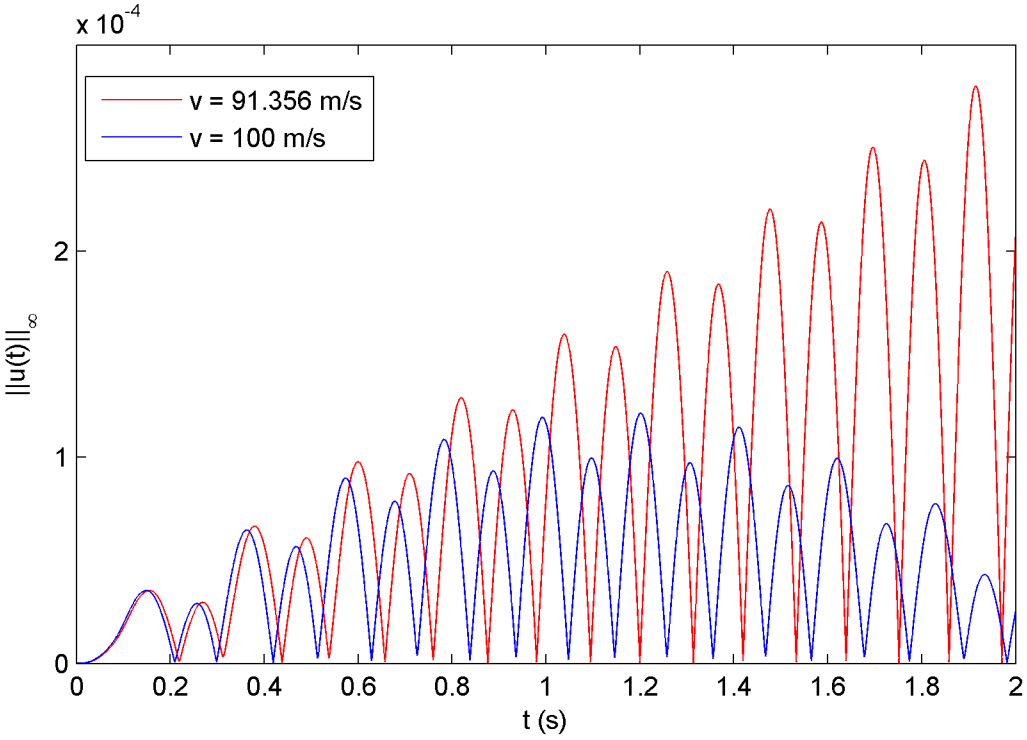


Figure 8.9: Resonance occurring when $v \approx 91.356$m/s.

As one can see, the two solutions are similar at the beginning. However, after a while the two solutions become out of phase. For the case where $v = 100$m/s, the phase change between the exerted pressure and the first eigenfrequency will create damping. However, for the critical speed $v = 91.356$, the periods are equal and hence resonance occurs.

8.8 Conclusion

Two methods have been proposed in order to solve the beam equation (8.4). First, there is the usual finite difference approach. By discretising the bridge into elements and approximating the derivatives we are capable of solving both the static as well as the time-dependent beam equation. In the latter case, a time integration scheme should be used. We have shown that explicit integration schemes for this differential equation can suffer from a CFL condition. This CFL condition is fairly strict, and stability is only ensured if $\Delta t = \mathcal{O}(\Delta x^2)$. Spatial accuracy is therefore limited since the number of time iterations need to increase rapidly. This is however not the case for an implicit method such as Backward Euler.

Secondly, the modal approach has been introduced. The mode shapes and the corresponding eigenfrequencies can be computed analytically by the method of separation of variables. Alternatively, as shown in Section 8.4.4, they can also be approximated by looking at the eigenvectors and eigenvalues of the finite difference matrix. We have derived an ordinary differential equation for each mode shape. These differential equations are independent and can therefore be solved independently. Both explicit as well as implicit integration schemes can be applied. Additionally, each iteration requires the computation of an integral. This integral can be approximated by using an integrator such as the trapezoidal rule or a similar Newton-Cotes formula.

Both approaches have been applied to solve a basic moving load problem. The solutions computed using the finite difference method corresponds to the modal solution. We have shown (for the static case, the time-dependent case seem to have simular results) that the error of the modal approach is of order $\mathcal{O}(m^{-3})$ so that the solution converges quickly. Usually, taking 50 mode shapes is enough to get accurate results.

The modal approach looks promising. It is a stable and computationally inexpensive approach of solving the beam equation. Additionally, each modal coefficient can be solved independently of each other. Because of these advantages, we will use this approach in the coming chapters.

Chapter 9

Combining local and global deformations using CONTACT

In the previous chapter we have introduced the concept of the global deformation of a beam. This deformation is the result of a pressure being exerted on the surface. However, as we have discussed in Chapter 6 and 7, this pressure will also result in local deformation, i.e. deformation strictly around the contact area.

The finite difference approach can be used to solve the 2 dimensional (i.e. in the x and z direction) linear elasticity equations given the right boundary conditions. This approach will automatically take both global and local deformation into account. However, as we have discussed in Chapter 7, this approach is computationally very expensive, often unstable and the results do not always seem realistic.

The goal of this chapter is to show how CONTACT can be used to combine local deformation with the global deformation of a bridge by using mode shapes. This approach will be computationally much less expensive than the complete finite difference approach. Some assumptions need to be made which will be discussed in this chapter.

9.1 Assumptions

9.1.1 The total deformation of a beam

We construct the total deformation a beam as the sum of the global and local deformation, i.e.

$$u_{\text{tot}}(x,t) = u(x,t) + l(x,t) \tag{9.1}$$

where u_{tot} is the total deformation. u denotes the global deformation, i.e. the solution computed using modes, and l the local deformation. The local deformation must have the important property that for fixed t, l is (almost) zero almost everywhere, except near the area of contact.

9.1.2 The penetration and approach for a globally deformed beam

In Chapter 6 we have discussed the basic dynamics of a rigid sphere falling on an elastic half-space. We have seen the approach δ is simply given by $\delta(t) = -z(t)$, where $z(t)$ is the height of the lowest point of the sphere. Similarly, the penetration $\delta(x, y, t)$ is given by 6.10. This example is simply since there is no global deformation.

However, consider the same sphere falling on an elastic beam. The beam is much more flexible than the half-plane since elements on the bottom boundary are not fixed. As a result, the height $-z(t)$ does not equal to the approach $\delta(t)$. Instead, we assume that the penetration is defined with respect to the globally deformed beam, i.e.

$$\delta(x, y, t) = u(x, t) - z(x, y, t) \tag{9.2}$$

where $z(x, y, t)$ is the height of the surface of the sphere at position (x, y) at time t. The approach δ is as usual $\delta(t) = \max_{(x,y)\in\Omega}\delta(x, y, t)$ the maximum penetration.

Using this definition, the two problems are consistent in that the approach and penetration is computed with respect to the global deformed situation (which is zero for the half-plane problem). The assumption is realistic since global deformation is more of a result of bending and vertical translation, the material itself is not really compressed.

9.1.3 The pressure for a globally deformed beam

Since the penetration between the object and the beam is now defined with respect to the global deformation of the beam, the situation becomes more complex. Hertz theory to compute the pressure distribution and normal force is no longer valid since the geometry of the surface of the beam is no longer flat as was the case for the half-space.

As discussed in Section 4.5, CONTACT can be used for arbitrarily shaped objects. CONTACT returns the pressure distribution $p(x, y)$, the normal force F_n, as well as the static-deformation. The same pressure distribution should be used to solve the beam equation.

9.1.4 The quasi-static local deformation

One of the advantages of the modal approach is that we do not need to discretise with respect to the z direction, resulting in a much less computationally expensive algorithm. However, this comes with a disadvantage. Without discretising in the z direction, the shock behaviour occurring during and after the impact of an object on a half-plane or bridge can not be computed and combined with the modal approach for the global deformation.

Therefore, we will make the important assumption that this shock behaviour can be neglected. This shock will only occur near the contact area. This area is for the train-bridge problem very small. By using the static local deformation $\tilde{u}_l$ computed using CONTACT for the local deformation, we are effectively solving the quasi-static elasticity equations. Using this will result in an error, but we expect this error to be small, especially compared to the global deformation of the bridge.

9.2 Combining global and local deformations

In Chapter 6 we discussed the physics involving a sphere falling on an elastic surface. Using Hertz theory we derived a differential equation that described the height of a rigid sphere falling on an elastic half-plane. In Chapter 8 we described the total deformation and bending of a beam.

These two chapters describe different phenomena. In Chapter 6 we only consider *local* deformation, the elastic surface itself does not move. We derived a differential equation and algorithm to compute the height and quasi-static local deformation of the half-plane. Chapter 8 is focussed on the *global* deformation that occurs when there are forces being exerted on the beam.

The interesting and challenging problem is that the two different kinds of deformation are not independent of each other. Using mode shapes, the global deformation can be computed if the external pressure p is known. This pressure distribution around the contact area can be computed using CONTACT when supplying the penetration. This penetration, however, is the distance between the sphere and the global deformed bridge. This information is not yet known since the goal is to compute this global deformation. A very similar problem involving the height of the sphere also occurs. This height depends on the normal force of the bridge which is computed using CONTACT after supplying the penetration. But this penetration is not known since it is dependent on the height of the sphere.

Computing the total deformation u_{tot} of the bridge is therefore not straightforward. An algorithm should be used to compute the pressure distribution p, global deformation u, as well as the contact force F_n in such a way both the global and local problems are consistent with each other. That is, if the pressure is recomputed using the already computed global deformation w and normal force F_n, the exact pressure distribution should be derived.

9.3 The stationary problem

For simplicity, we consider the stationary problem of a train standing still on a bridge. Our goal is to compute the total deformation of the bridge $u_{\text{tot}}(x)$. To do so accurately, both the global and local phenomena should be taken into account. Denote the height of the train wheels with respect to the xy-plane by $w(x)$. The undeformed distance, similarly to the penetration for the instationary case, between the train and the locally undeformed bridge is therefore given by

$$h(x) = u(x) - w(x) \tag{9.3}$$

Since we are looking at the stationary solution, we know that the normal force is equal to $F_n = -F_g = m_c g$, where m_c is the mass of the cylinder. This normal force as well as the undeformed distance can then be supplied to CONTACT. CONTACT will then the pressure distribution $p(x)$ around the contact area as well as the quasi-static local deformation $l(x)$.

Note that $p(x)$, computed by CONTACT, can only be computed after computing the global deformation $u(x)$. This global deformation, on the other hand, depends on the same pressure distribution $p(x)$ since it is the solution of the stationary beam equation

$$EI\frac{\mathrm{d}^4u(x)}{\mathrm{d}x^4} = p(x) \tag{9.4}$$

The problem that we see here is that the pressure distribution $p(x)$ depends on the global deformation $u(x)$, which depends on $p(x)$. Since the computation from u to p using CONTACT is one way only, i.e. this step cannot be inversed, $p(x)$ cannot be solved explicitly.

The goal therefore is to compute the pressure distribution $p(x)$ and the global deformation $g(x)$ of the bridge by using an implicit solver. See Algorithm 9.1.

Algorithm 9.1 Computing the total static deformation of a beam with CONTACT

1: Set $u(x) \equiv 0$.
2: **repeat**
3: Set $u^* \equiv u$.
4: Set $h(x) = u^*(x) - w(x)$ for all x.
5: Compute $p(x)$ and $l(x)$ with CONTACT using normal force $F_n = m_c g$ and undeformed distance h.
6: **for** $i = 1$ to m **do** ▷ Computing modal coefficients using (8.49)
7: Compute $c_i = \dfrac{1}{\rho\lambda_i^2}\int_0^L p(x)w_i(x)\mathrm{d}x$ using a Newton-Cotes formula.
8: **end for**
9: Set $u(x) = \sum_{i=1}^m c_i w_i(x)$ for all x.
10: Set error $= \|u - u^*\|_2$.
11: **until** error $\leq \varepsilon$
12: Set $u_{\mathrm{tot}}(x) = u(x) + l(x)$.

If the algorithm converges, then the pressure distribution $p(x)$ both corresponds to the global problem (the beam equation) as well as the local problem (computed using CONTACT). The stopping criterion that checks for the difference in global deformation u, i.e. $\|u - u^*\|_2 < \varepsilon$, could also be changed check for a difference in pressure distribution, i.e. convergence is reached if and only if $\|p - p^*\| < \varepsilon$.

Numerically it can be checked to see that the algorithm usually converges, as long as the global deformation is not extraordinary large (that is; the maximum vertical displacement shouldn't be more than 5% of the length of the bridge). Additionally, it is important that the mesh used to discretise the penetration for CONTACT should be fine enough; using 20×20 grid points around the contact area is more than enough. Otherwise, convergence might not be reached.

See Figure 9.1 for the result. Note that the ratios in this figure are not realistic, the parameters are chosen so that the result is more visually clear. The purple line represents the global deformation, the red line the global deformation. The total deformation is the sum of the local and global deformation. One can see that the wheel perfectly fits within the local deformed bridge.

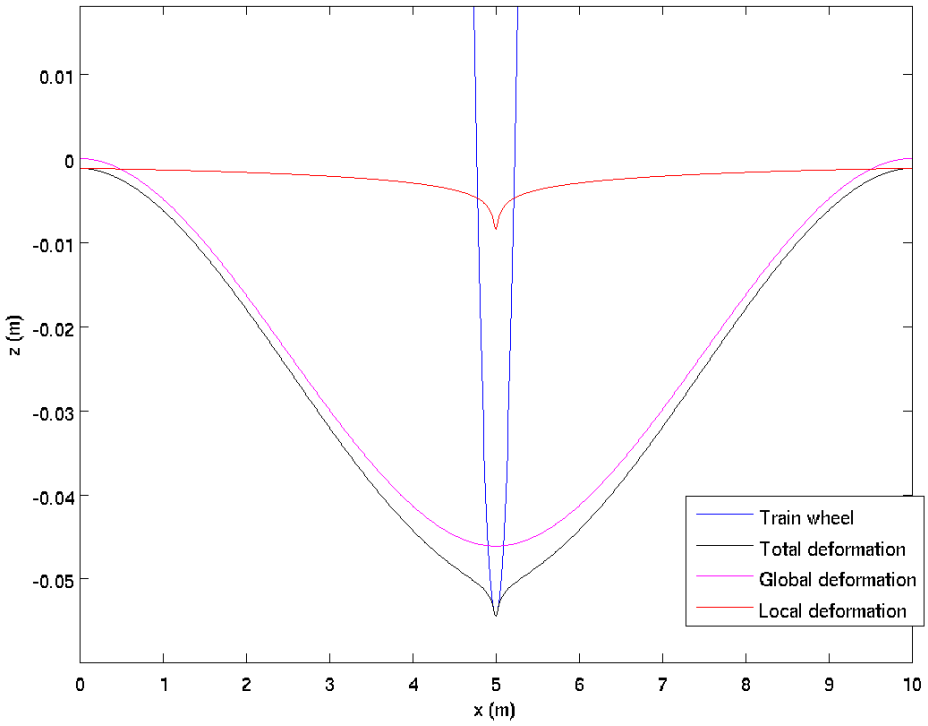


Figure 9.1: A visual interpretation of a train wheel on a bridge, combining global and local deformations

9.4 The time-dependent problem

Consider the same problem, but now suppose that the height of the wheel $w(x,t)$ is also a function of time. To make it slightly easier, we suppose this height is known beforehand and unrelated to the contact physics between the wheel and the bridge. In Chapter 10 we will discuss the most general case where the height of the wheel depends on the current state of the bridge.

The penetration at a given time t is given by

$$\delta(x,t) = u(x,t) - w(x,t) \tag{9.5}$$

The global deformation u is constructed by using modes. The modal coefficients $c_i(t)$ for $i = 1, \ldots, m$ are computed by applying a numerical integration scheme to solve differential equation (8.59). This scheme can be either explicit or implicit, which this is of importance for the algorithm.

9.4.1 Explicit integration schemes

For explicit schemes, the problem is easy. When computing the global deformation at time step j, the global deformation as time step $j-1$ is used. This information is known so that we can compute the pressure distribution p and the local deformation l explicitly. See Algorithm 9.2.

Algorithm 9.2 Computing the total dynamic deformation of a beam with CONTACT with an explicit integration scheme

1: Set $c_i^0 = \dot{c}_i^0 = 0$ for all $1 \leq i \leq m$. If the initial condition is not zero, use (8.61) and (8.62) instead.
2: Set $u^0(x) \equiv 0$.
3: **for** $k = 1$ to n **do** ▷ Loop over each of n time steps
4: Set $\delta(x) = u^{k-1}(x) - w(x,t)$ for all x.
5: Compute $p(x)$ and $l(x)$ with CONTACT using penetration $\delta(x)$.
6: **for** $i = 1$ to m **do** ▷ Computing modal coefficients using (8.63)
7: Apply the explicit integration scheme to compute c_i^k for

$$c_i'(t) = \dot{c}_i(t)$$

8: Apply a Newton-Cotes formula and the explicit integration scheme to compute $\dot{c}_i^k$ for

$$\dot{c}_i'(t) = \frac{1}{\rho}\int_0^L p(x,t)w_i(x)\mathrm{d}x - \frac{EI}{\rho}\beta_i^4 c_i(t)$$

9: **end for**
10: Set $u^k(x) = \sum_{i=1}^m c_i^k w_i(x)$ for all x.
11: Set $u_{\text{tot}}^k(x) = u^k(x) + l(x)$ for all x.
12: **end for**

However, we have seen in Chapter 6 that explicit integration schemes are generally not usable if the material of the half-space (or bridge in this situation) is stiff. The use of this algorithm is therefore limited.

9.4.2 Implicit integration schemes

For implicit time integration methods such as Radau5, then at each time step we have a similar problem as in the stationary case. The global deformation at time step j can not be computed without knowing the pressure p at the same time step, which is also not known. A combination between Algorithm 9.1 and 9.2 is therefore proposed, see Algorithm 9.3.

Algorithm 9.3 Computing the total dynamic deformation of a beam with CONTACT with an implicit integration scheme

1: Set $c_i^0 = \dot{c}_i^0 = 0$ for all $1 \leq i \leq m$. If the initial condition is not zero, use (8.61) and (8.62) instead.
2: Set $u^0(x) \equiv 0$.
3: **for** $k = 1$ to n **do** ▷ Loop over each time step
4: Set $u \equiv u^{k-1}$.
5: **repeat**
6: Set $u^* \equiv u$.
7: Set $\delta(x) = u(x) - w(x, t^k)$ for all x.
8: Compute $p(x)$ and $l(x)$ with CONTACT using penetration $\delta(x)$.
9: **for** $i = 1$ to m **do** ▷ Computing modal coefficients using (8.63)
10: Apply the integration scheme explicitly (using $p(x), c_i^{k-1}$ and $\dot{c}_i^{k-1}$ in place for $p^k(x), c_i^k$ and $\dot{c}_i^k$ respectively) to compute c_i^j for

$$c_i'(t) = \dot{c}_i(t)$$

11: Apply a Newton-Cotes formula and the integration scheme explicitly to compute $\dot{c}_i^k$ for

$$\dot{c}_i'(t) = \frac{1}{\rho} \int_0^L p(x,t) w_i(x) \mathrm{d}x - \frac{EI}{\rho} \beta_i^4 c_i(t)$$

12: **end for**
13: Set $u^k(x) = \sum_{i=1}^m c_i^k w_i(x)$ for all x.
14: Set error $= \|u - u^*\|_2$.
15: **until** error $\leq \varepsilon$
16: Set $u_{\text{tot}}^k(x) = u^k(x) + l(x)$ for all x.
17: **end for**

If this algorithm converges, then $u^* \approx u$ for each time step and hence c_i satisfies the implicit expression of the time integration scheme for all $1 \leq i \leq m$.

Chapter 10

Full train-bridge simulation

In this chapter, we will develop an algorithm to be able to perform an accurate train-bridge simulation by using CONTACT. This algorithm makes use of the theory as discussed in the previous chapters. The numerical methods as described in Chapter 5 will be used to integrate the differential equations for the modal coefficients as derived in Section 8.4. The algorithm works closely together with CONTACT in order to compute the pressure distribution around the point of contact as well as the normal contact force accurately.

In Chapter 9 we have seen that a fairly simple Picard iteration can be used to combine the local and global contact problem using CONTACT. However, we assumed in this problem the height of the wheel $u_w(x,t)$ is known beforehand. In reality, this height satisfies a similar differential equation as the one described in Chapter 6. This differential equation, however, strongly depend on each of the multiple differential equations for the mode shapes. The new contact problem turns out to be much more complex. The iterative Picard iteration can be adapted but will turn out to break down very quickly. Therefore, we had to figure out a way to improve the stability of the iterative solver.

10.1 Formulation of the problem

We consider a single cylindrical wheel of radius R moving over a bridge of length L with a horizontal speed of w. The bridge is 20m long, 8m width, 50cm thick and is made of steel. The train wheel is also made of steel and can deform locally. See Table 10.1 for the corresponding values.

The global deformation of the bridge is denoted by $u(x,t)$ and is approximated by m modes. The geometry of the wheel is denoted by $g(x)$ and is defined as the difference in height of the wheel with respect to its lowest point, the reference point. For a cylindrical wheel of radius R, it is given by

$$g(x) = R - \sqrt{R^2 - x^2} \qquad \text{for } |x| \leq R \tag{10.1}$$

However, we do allow arbitrary functions of g, i.e. arbitrarily shaped wheels, as long as g is non-negative and zero at the reference point. The actual time-dependent height of the wheel,

Property	Value	Explanation
I	0.0104m^4	$I(x) = \int_{-0.25}^{0.25} \lvert z\rvert^2 \mathrm{d}z$
L	20m	Length of bridge
B	8m	Width of bridge
s	30m/s	Speed of train in m/s
E	$2 \cdot 10^{11}$	Elastic modulus of steel
ρ	7900kg/m^3	Density of steel
R	0.3m	Radius of wheel
m_c	$76 \cdot 10^4\text{kg}$	Mass of cylinder

Table 10.1: The properties of the bridge and the train wheel.

measured from the same lowest point, is denoted by $z(t)$. The wheel moves at a constant speed of s, so that the actual height of the undeformed wheel w_{tot} is given by

$$w_{\text{tot}}(x,t) = g(x - x_0 - st) + z(t) \qquad \text{for } |x| \leq st \tag{10.2}$$

where x_0 is the starting position of the reference element on the wheel.

Both the wheel as well as the bridge are capable of deforming locally. The local deformation $l(x)$ occurring during the contact is computed as the quasi-static result of CONTACT. Since both the wheel as the bridge are made of the same material, no friction occurs and the local deformation is evenly spread among the two objects, i.e.

$$\begin{aligned} u_{\text{tot}}(x,t) &= u(x,t) + \frac{1}{2}l(x) \\ w_{\text{tot}}(x,t) &= w(x,t) - \frac{1}{2}l(x) \end{aligned} \tag{10.3}$$

The penetration $\delta(x,y,t)$ is defined as the distance between the wheel and the globally deformed bridge at time t, a positive value representing penetration. Since the problem is independent of the y axis (both the height of the wheel as well as the height of the bridge is constant with respect to y), this penetration is actually a function of x and t only. It is by definition equal to

$$\delta(x,t) = u(x,t) - w(x,t) = u(x,t) - z(t) - g(x - x_0 - st) \tag{10.4}$$

The normal force F_n as well as the pressure distribution p is a function of $\delta(x,t)$ and can be computed by using CONTACT. The normal force F_n directly influences the height of the wheel. By the second law of Newton, the height $z(x)$ satisfies

$$z''(t) = \frac{F_n}{m_c} - g \tag{10.5}$$

For the bridge, the modal coefficients $c_j(t)$ satisfy the modal equation

$$EIc_j(t)\beta_j^4 + \rho c_j''(t) = \int_0^L p(x,t)w_j(x)\mathrm{d}x \tag{10.6}$$

Differential equation (10.5) is dependent on each of the modal differential equations (10.6). This is because the contact force F_n is computed using CONTACT by supplying the penetration $\delta(x,t)$. This penetration is constructed using $u(x,t) = \sum_{i=1}^{m} c_i(t)w_i(x)$ and hence the dependency on $c_i(t)$. This relation also works in the opposite direction; The pressure $p(x,t)$ of (10.6) depends on the approach $\delta(t)$ and hence also on $z(t)$.

10.2 The system of differential equations

To solve the problem, equations (10.5) and (10.6) should be combined to get one single differential equation. To do so, define the vector $\mathbf{v}(t)$ by

$$v_i(x,t) = \begin{cases} c_i & \text{for } 1 \leq i \leq m \\ z & \text{for } i = m+1 \end{cases} \tag{10.7}$$

The penetration and contact force can then be noted by $p(x,\mathbf{v},t)$ and $F_n(\mathbf{v},t)$ respectively, so emphasize the dependency on both the modal coefficients c_j as well as the height of the cylinder z.

Both differential equation can be combined to get

$$\begin{aligned} v_1''(t) &= \frac{-EI\beta_1^4}{\rho} v_1(t) + \frac{1}{\rho}\int_0^L p(x,\mathbf{v},t)w_1(x)\mathrm{d}x \\ &\vdots \qquad\qquad \vdots \\ v_m''(t) &= \frac{-EI\beta_m^4}{\rho} v_m(t) + \frac{1}{\rho}\int_0^L p(x,\mathbf{v},t)w_m(x)\mathrm{d}x \\ v_{m+1}''(t) &= \frac{F_n(\mathbf{v},t)}{m_c} - g \end{aligned} \tag{10.8}$$

which can also be denoted as

$$\ddot{\mathbf{v}}(t) = A\mathbf{v}(t) + \mathbf{f}(\delta(\mathbf{v},t)) \tag{10.9}$$

10.3 Implementation of Backward Euler

Next, we will apply the Backward Euler integration scheme to try to solve system (10.8). The reason for this choice is that Backward Euler is the simplest implicit integration scheme. As we will see later, the system can become very complex. Backward Euler will therefore be used to illustrate the problem, but any other time integration scheme as discussed in Chapter 5 can also be used instead.

The Backward Euler integration scheme can only be applied to first-order ordinary differential equation. Hence, we introduce components $\dot{c}_i(t)$ for $1 \leq i \leq m$ as the modal speed components and $\dot{z}$ as the vertical speed of the cylinder. System (10.8) can then be rewritten as the first order ordinary differential equation

$$\begin{cases} c_1'(t) = \dot{c}_1(t) \\ \dot{c}_1'(t) = \dfrac{-EI\beta_1^4}{\rho} c_1(t) + \dfrac{1}{\rho}\displaystyle\int_0^L p(x, \mathbf{c}, z, t) w_1(x) \mathrm{d}x \\ \quad \vdots \qquad\qquad \vdots \\ c_m'(t) = \dot{c}_m(t) \\ \dot{c}_m'(t) = \dfrac{-EI\beta_m^4}{\rho} c_m(t) + \dfrac{1}{\rho}\displaystyle\int_0^L p(x, \mathbf{c}, z, t) w_m(x) \mathrm{d}x \\ z'(t) = \dot{z}(t) \\ \dot{z}'(t) = \dfrac{F_n(\mathbf{c}, z, t)}{m_c} - g \end{cases} \tag{10.10}$$

Backward Euler applied to (10.10) is defined by

$$\begin{cases} c_1^{k+1} = c_1^k + \Delta t \dot{c}_1^{k+1} \\ \dot{c}_1^{k+1} = \dot{c}_1^k + \dfrac{\Delta t}{\rho}\left[\displaystyle\int_0^L p(x, \mathbf{c}^{k+1}, z^{k+1}, t^{k+1}) w_1(x) \mathrm{d}x - EI\beta_1^4 c_1^{k+1}\right] \\ \quad \vdots \qquad\qquad \vdots \\ c_m^{k+1} = c_m^k + \Delta t \dot{c}_m^{k+1} \\ \dot{c}_m^{k+1} = \dot{c}_m^k + \dfrac{\Delta t}{\rho}\left[\displaystyle\int_0^L p(x, \mathbf{c}^{k+1}, z^{k+1}, t^{k+1}) w_m(x) \mathrm{d}x - EI\beta_m^4 c_m^{k+1}\right] \\ z^{k+1} = z_k + \Delta t \dot{z}^{k+1} \\ \dot{z}^{k+1} = \dot{z}^k + \Delta t\left[\dfrac{F_n(\mathbf{c}^{k+1}, z^{k+1}, t^{k+1})}{m_c} - g\right] \end{cases} \tag{10.11}$$

The terms $\Delta t \dot{c}_i^{k+1}$, $\Delta t \dot{z}^{k+1}$, and $\dfrac{-EI\beta_i^4}{\rho} c_i^{k+1}$ for $1 \leq i \leq m$ can be moved to the left-hand side to get

$$\begin{cases} c_1^{k+1} - \Delta t \dot{c}_1^{k+1} = c_1^k \\ \dot{c}_1^{k+1} + \Delta t \dfrac{EI}{\rho}\beta_1^4 c_1^{k+1} = \dot{c}_1^k + \dfrac{\Delta t}{\rho}\displaystyle\int_0^L p(x, \mathbf{c}^{k+1}, z^{k+1}, t^{k+1}) w_1(x) \mathrm{d}x \\ \qquad\qquad \vdots \qquad\qquad \vdots \\ c_m^{k+1} - \Delta t \dot{c}_m^{k+1} = c_m^k \\ \dot{c}_m^{k+1} + \Delta t \dfrac{EI}{\rho}\beta_m^4 c_m^{k+1} = \dot{c}_m^k + \dfrac{\Delta t}{\rho}\displaystyle\int_0^L p(x, \mathbf{c}^{k+1}, z^{k+1}, t^{k+1}) w_m(x) \mathrm{d}x \\ z^{k+1} - \Delta t \dot{z}^{k+1} = z_k \\ \dot{z}^{k+1} = \dot{z}^k + \Delta t\left[\dfrac{F_n(\mathbf{c}^{k+1}, z^{k+1}, t^{k+1})}{m_c} - g\right] \end{cases} \tag{10.12}$$

or in matrix-vector notation

$$\begin{pmatrix} 1 & -\Delta t & & & & & & \\ \Delta t\frac{EI}{\rho}\beta_1^4 & 1 & & & & & & \\ & & \ddots & & & & & \\ & & & 1 & -\Delta t & & & \\ & & & \Delta t\frac{EI}{\rho}\beta_m^4 & 1 & & & \\ & & & & & 1 & -\Delta t & \\ & & & & & 0 & 1 & \end{pmatrix} \begin{pmatrix} c_1 \\ \dot{c}_1 \\ \vdots \\ c_m \\ \dot{c}_m \\ z \\ \dot{z} \end{pmatrix}^{k+1} = \begin{pmatrix} c_1 \\ \dot{c}_1 \\ \vdots \\ c_m \\ \dot{c}_m \\ z \\ \dot{z} \end{pmatrix}^{k} + \Delta t \mathbf{f}(\mathbf{c}^{k+1}, z^{k+1}, t^{k+1}) \tag{10.13}$$

where

$$\mathbf{f}(\mathbf{c}^{k+1}, z^{k+1}, t^{k+1}) = \begin{pmatrix} 0 \\ \frac{1}{\rho}\int_0^L p(x, \mathbf{c}^{k+1}, z^{k+1}, t^{k+1}) w_1(x) \mathrm{d}x \\ \vdots \\ 0 \\ \frac{1}{\rho}\int_0^L p(x, \mathbf{c}^{k+1}, z^{k+1}, t^{k+1}) w_m(x) \mathrm{d}x \\ 0 \\ \frac{F_n(\mathbf{c}^{k+1}, z^{k+1}, t^{k+1})}{m_c} - g \end{pmatrix} \tag{10.14}$$

10.4 The Picard approach

10.4.1 The algorithm

Due to term $\mathbf{f}(\mathbf{c}^{k+1}, z^{k+1}, t^{k+1})$ on the right-hand side, expression (10.13) is implicit. The process of computing the normal force as well as the pressure distribution using CONTACT is one-way only, i.e. we are not capable of computing the inverse of this process. Hence, an iterative solver must be used at each time step in order to solve equation (10.13).

We will introduce a new index j which represents the iteration number. Furthermore, the components z_j^{k+1} and $c_{i,j}^{k+1}$ for $1 \leq i \leq m$ represent the approximations of z^{k+1} and c_i^{k+1}, respectively. The goal is to define z_j^{k+1} and $c_{i,j}^{k+1}$ iteratively so that

$$\begin{cases} z_j^{k+1} \to z^{k+1} \\ c_{i,j}^{k+1} \to c_i^{k+1} \text{ for } 1 \leq i \leq m \end{cases} \qquad \text{as } j \to \infty \tag{10.15}$$

Preferably, the convergence should be quick since the computation of the normal force and pressure distribution using CONTACT (as well as the communication between CONTACT and Matlab, done by writing/reading input/ouput files) is computationally very expensive compared to other computations such as solving a small linear system using Matlab.

The straightforward approach is to apply a Picard iteration. For the best stability of the iterative process, it is better to take the linear term $\frac{-EI\beta_i^4}{\rho}c_i^{k+1}$ as in (10.12) implicit. p and

F_n need to be explicit terms in the iteration since they rely on the results of CONTACT. Each iteration therefore involves solving the linear system as in (10.13) with the right-hand side being computed explicitly, i.e. at each iteration we compute

$$\begin{pmatrix} c_1 \\ \dot{c}_1 \\ \vdots \\ c_m \\ \dot{c}_m \\ z \\ \dot{z} \end{pmatrix}_{j+1}^{k+1} = \begin{pmatrix} 1 & -\Delta t & & & & & \\ \Delta t \frac{EI}{\rho}\beta_1^4 & 1 & & & & & \\ & & \ddots & & & & \\ & & & 1 & -\Delta t & & \\ & & & \Delta t \frac{EI}{\rho}\beta_m^4 & 1 & & \\ & & & & & 1 & -\Delta t \\ & & & & & 0 & 1 \end{pmatrix}^{-1} \left[\begin{pmatrix} c_1 \\ \dot{c}_1 \\ \vdots \\ c_m \\ \dot{c}_m \\ z \\ \dot{z} \end{pmatrix}^{k} + \Delta t \mathbf{f}(\mathbf{c}_j^{k+1}, z_j^{k+1}, t^{k+1}) \right] \tag{10.16}$$

A convergence criterion is to check the relative difference between $\mathbf{v}_{j+1}^{k+1}$ and $\mathbf{v}_j^{k+1}$ in a certain norm. For example, we can repeat this process at each time step until $\|\mathbf{v}_{j+1}^{k+1} - \mathbf{v}_j^{k+1}\|_2 / \|\mathbf{v}_j^{k+1}\|_2 < \varepsilon$ for some small value of ε.

Algorithm 10.1 Performing a full train-bridge simulation using Picard iteration as solver

1: Set $c_i^0 = \dot{c}_i^0 = 0$ for all $1 \leq i \leq m$. If the initial condition is not zero, use (8.61) and (8.62) instead.
2: Set $z^0 = \dot{z}^0 = 0$ or equal to the corresponding initial conditions.
3: Set $u^0(x) \equiv \sum_{i=1}^m c_i^0 w_i(x)$.
4: **for** $k = 1$ to n **do**
5: Set $\mathbf{y} = [c_1^{k-1}; \dot{c}_1^{k-1}; \ldots; c_m^{k-1}; \dot{c}_m^{k-1}; z^{k-1}; \dot{z}^{k-1}]$.
6: **repeat**
7: Set $\mathbf{y}^* = \mathbf{y}$.
8: Set $\delta(x) = \sum_{i=1}^m c_i^* w_i(x) - z^* - g(x - x_0 - st^k)$ for all x.
9: Compute $p(x)$, F and $l(x)$ with CONTACT using penetration $\delta(x)$.
10: Compute right-hand vector $\mathbf{g} = \mathbf{y}^{k-1} + \Delta t \mathbf{f}(\mathbf{y}^*, t^k)$ as in (10.13), using a Newton Cotes formula for $\mathbf{f}$.
11: Solve (10.13) for $\mathbf{y} = [c_1; \dot{c}_1; \ldots; c_m; \dot{c}_m; z; \dot{z};]$.
12: Set error $= \|\mathbf{y} - \mathbf{y}^*\|_2$.
13: Set $\mathbf{y} = (1 - \omega)\mathbf{y}^* + \omega \mathbf{y}$
14: **until** error $\leq \varepsilon$
15: Set $u^k(x) = \sum_{i=1}^m c_i w_i(x)$ for all x
16: Set $u_{\text{tot}}^k(x) = u^k(x) + \frac{1}{2} l(x)$ for all x.
17: Set $w^k(x) = g(x - x_0 - st^k) + z$ for all x.
18: Set $w_{\text{tot}}^k(x) = w^k(x) - \frac{1}{2} l(x)$ for all x.
19: **end for**

Equation (10.16) can be written as $\mathbf{y}_{j+1}^{k+1} = A^{-1}(\mathbf{y}^k + \Delta t \mathbf{f}(\mathbf{c}_j^{k+1}, z_j^{k+1}, t^{k+1}))$. To achieve better convergence, we will also introduce the principle of under relaxation. The relaxation factor $\omega > 0$ is a value that represents the magnitude of the relaxation. If $\omega < 1$, then this is called under relaxation; $\omega > 1$ represents over relaxation. The idea is that a single Picard iteration might overshoot so that the sequence $(\mathbf{y}_j^{k+1})_{j=1}^{\infty}$ diverges. To prevent this, we can instead set

$$\mathbf{y}_{j+1}^{k+1} = (1-\omega)\mathbf{y}_j^{k+1} + \omega A^{-1}(\mathbf{y}^k + \Delta t \mathbf{f}(\mathbf{c}_j^{k+1}, z_j^{k+1}, t^{k+1})) \tag{10.17}$$

If $\omega = 1$, there is no under relaxation and hence (10.17) is the same as (10.16). If ω is close to 0, then the difference between $\mathbf{y}_{j+1}^{k+1}$ and $\mathbf{y}_j^{k+1}$ is small. This can prevent overshoot in the Picard process, thereby improving the stability of the method. The downfall of this is that single time step might take more iterations, so more computational power is required. This will be shown below.

The actual height of the wheel and bridge at time step t^k is represented by $w_{\text{tot}}^k(x)$ and $u_{\text{tot}}^k(x)$. These values include the rigid displacements of the objects as well as the total deformation. See Algorithm 10.1.

10.4.2 Numerical results

Running CONTACT is computationally fairly expensive. On an average modern computer, a single CONTACT run takes about half a second, or more if a finer mesh is given for a better precision. The Picard iteration might take 10 steps until convergence is reached, which would make a single time step to take about 5 seconds. The number of time steps we can do is therefore very limited. In practice we were not able to use a time step smaller than $\Delta t = 0.01$s and still being able to perform a meaningful simulation.

Using this fixed time step and the properties as in Table 10.1, the Picard iteration does not converge. To understand this behaviour, it is important to note that the matrix as in (10.16) is a block diagonal matrix. Between the transition from iteration j to $j+1$, there is no interaction between the mode coefficients and the height of the wheel nor any interaction between the mode coefficients mutually. As an example, suppose that the wheel at iteration j has height 0m and vertical speed -0.1m/s. The height and vertical speed of the wheel at iteration $j+1$ using a time step $\Delta t = 0.01$ will be given by

$$\begin{pmatrix} z_{j+1}^{k+1} \\ \dot{z}_{j+1}^{k+1} \end{pmatrix} = \begin{pmatrix} 1 & -\Delta t \\ 0 & 1 \end{pmatrix}^{-1} \begin{pmatrix} 0 \\ -0.1 - \Delta t g \end{pmatrix} \approx \begin{pmatrix} -.002 \\ -.198 \end{pmatrix} \tag{10.18}$$

Hence, after this iteration, the approach is roughly 0.2mm. For a rigid material like steel, this is huge. The corresponding contact force (assuming no global deformation) is enormous and is approximately $F_n = 9.6 \cdot 10^8 N$. The next iteration will try to correct this error but this will only results in an even larger error:

$$\begin{pmatrix} z_{j+2}^{k+1} \\ \dot{z}_{j+2}^{k+1} \end{pmatrix} = \begin{pmatrix} 1 & -\Delta t \\ 0 & 1 \end{pmatrix}^{-1} \begin{pmatrix} -.002 \\ -.198 + \Delta t(\frac{F_n}{m_c} - g) \end{pmatrix} \approx \begin{pmatrix} 5.274 \\ 527.6 \end{pmatrix} \tag{10.19}$$

The extreme approach resulted in an extreme correction by the Picard iteration; the wheel was suddenly hovering 5 meters above the bridge. The iteration will clearly not converge.

It is still worth checking when the problem does converge and how we can improve the stability. It appears that if a very low Young's modulus is used for the the bridge, for example if we consider the bridge to be made out of rubber, the algorithm converges. Our goal is to maximize this Young's modulus E so that the Picard iteration converges. To do so, we will make use of under relaxation to prevent overshoot. We will count the number of iteration it takes until convergence is reached, i.e. $\|\mathbf{y} - \mathbf{y}^*\| < \varepsilon$ for $\varepsilon = 10^{-4}$ for the first time

step of the problem. In this experiments, the parameters as in Table 10.1 are used, except for the Young's modulus E which we will vary.

		E				
		$1 \cdot 10^6$	$5 \cdot 10^6$	$10 \cdot 10^6$	$20 \cdot 10^6$	$50 \cdot 10^6$
ω	1	6	16	∞	∞	∞
	0.9	4	12	∞	∞	∞
	0.7	8	7	13	∞	∞
	0.5	13	11	11	18	∞
	0.3	25	20	21	22	∞
	0.1	82	64	67	70	∞

Table 10.2: The number of iterations it takes until the Picard iteration converges.

In Table 10.2 the amount of time iterations are represented as function of E and ω. As we can see, under relaxation can have both positive and negative influence on the iterative process. For small E, using a slight under relaxation ω can lower the number of iterations needed. The larger E, the smaller ω should be in order to achieve convergence. In all situations, however, a smaller ω will require more iterations.

It is worth noting that Table 10.2 only shows the amount of iterations it takes until the iterative process for the first time step has converged. Often, the time integration can break down in the middle of the algorithm.

From Table 10.2 it is clear that under relaxation can be used to improve stability or even lower the number of iterations needed. However, the optimal choice of ω is different for different values of E. The use of under relaxation is limited. In practice, convergence cannot be reached for $E = 2 \cdot 10^{11}$ (i.e. the Young's modulus for steel) regardless of the (reasonable) choice of ω. Hence, we will need to improve our iterative solver.

10.5 The Quasi-Newton approach

10.5.1 Linearisation

The Newton-Raphson method is a well known root finding algorithm. In order to find a root of a function, the method effectively linearises the function around the current point and solves the resulting linear system. The method requires the derivative $f'(x_j)$ at each iteration, or the full Jacobi matrix $J(\mathbf{x}_j)$ with all partial derivatives for functions of multiple variables.

A similar root finder can be applied to equation (10.12) by moving all terms to the left hand side. The problem here is that the (partial) derivatives required by the Newton method are not known. For convenience, denote

$$F_j^{k+1} = F_n(\mathbf{c}_j^{k+1}, z_j^{k+1}, t^{k+1}), \text{ and} \qquad I_{i,j}^{k+1} = \int_0^L p(x, \mathbf{c}_j^{k+1}, z_j^{k+1}, t^{k+1}) w_i(x) \mathrm{d}x \tag{10.20}$$

Each of these terms relies on the results of CONTACT. CONTACT does not return time derivatives of F_n nor of the pressure p. The Newton method thus can not be applied. However,

approximations for each partial derivative can be made using finite differences. For example, we can approximate

$$\frac{\partial}{\partial z}F_j^{k+1} \approx \frac{F_n(\mathbf{c}_j^{k+1}, z_j^{k+1}, t^{k+1}) - F_n(\mathbf{c}_j^{k+1}, z_j^{k+1} - \alpha, t^{k+1})}{\alpha} \tag{10.21}$$

using backward differences. As long as CONTACT remains accurate enough, α can be chosen arbitrarily small so the low rate of convergence from forward differences is not an issue. Usually, we take $\alpha = 10^{-7}$. If, however, the approximation $\frac{\partial}{\partial z}F_j^{k+1}$ as in (10.21) turned out to be zero due to rounding issues and α being too small, even if penetration occurs at this iteration, then this is not a problem. In this case, we can dynamically increase α until the derivative is non-zero.

Using finite differences instead of the analytical derivatives for the Newton-Raphson method is a Quasi-Newton approach. The convergence of the this approach is only slightly slower than of the real Newton-Raphson method due to a small error in the derivative. In system (10.12), we have $2m + 2$ variables. Suppose that we apply Newtons method to solve (10.12). For multiple dimensions such as in this problem, linearising requires the computation of the Jacobi matrix at each iteration. This is, however, extremely expensive. Many derivatives, in particular $\frac{\partial}{\partial c_i}F_j^{k+1}$ for each $1 \leq i \leq m$, are not known analytically and should hence be approximated using finite differences. This is done in a similar way as (10.21). Each separate partial derivative requires a separate computation using CONTACT, so that we end up with over m CONTACT runs. This is extremely expensive. Additionally, this does not scale well with the number of modes m; the number of times CONTACT is ran should preferably not depend on m.

To solve this problem, one might argue not to linearise with respect to each and every variable and take some of the equations explicit. However, this can be dangerous since a small change in the geometry of the bridge can heavily alter the penetration $\delta(x, t)$ and therefore also the pressure as computed by CONTACT. Only linearising with respect to the height of the wheel $z(t)$, for example, is not enough; the iterative process will generally not converge. This is because we cannot neglect the difference in change of shape of the global deformed beam between two iterations j and $j + 1$.

The arguably most important variable in the system is the approach $\delta(t)$. The approach is the maximum penetration and can be formulated as

$$\begin{aligned}\delta(t) &= \max_{0\leq x\leq L}\left[u(x,t) - w(x,t)\right] \\ &= \max_{0\leq x\leq L}\left[\sum_{i=1}^{m} c_i(t)w_i(x) - z(t) - g(x - x_0 - st)\right] \\ &= \max_{0\leq x\leq L}\left[\sum_{i=1}^{m} c_i(t)w_i(x) - g(x - x_0 - st)\right] - z(t)\end{aligned} \tag{10.22}$$

The approach is therefore a fairly complex function dependent on each modal coefficient $c_i(t)$ as well as the height $z(t)$. We propose to linearise only with respect to the approach, so that we do not neglect any important information regarding the change of shape of the beam between two iterations.

The derivative with respect to the approach can be approximated in many different ways, but the most straightforward approach is

$$\frac{\partial}{\partial\delta}F_j^{k+1} = -\frac{\partial}{\partial z}F_j^{k+1} \approx -\frac{F_n(\mathbf{c}_j^{k+1}, z_j^{k+1}, t^{k+1}) - F_n(\mathbf{c}_j^{k+1}, z_j^{k+1} - \alpha, t^{k+1})}{\alpha}, \qquad \text{and}$$
$$\frac{\partial}{\partial\delta}I_{i,j}^{k+1} = -\frac{\partial}{\partial z}I_{i,j}^{k+1} \approx -\frac{\int_0^L (p(x, \mathbf{c}_j^{k+1}, z_j^{k+1}, t^{k+1}) - p(x, \mathbf{c}_j^{k+1}, z_j^{k+1} - \alpha, t^{k+1}))w_i(x)\mathrm{d}x}{\alpha} \tag{10.23}$$

Hence, by linearising F^{k+1} and I_i^{k+1} at each iteration j we arrive at the approximation

$$\begin{aligned} F_{j+1}^{k+1} &= F_j^{k+1} + \frac{\partial}{\partial\delta}F_j^{k+1} \cdot (\delta_{j+1}^{k+1} - \delta_j^{k+1}) \\ I_{i,j+1}^{k+1} &= I_{i,j}^{k+1} + \frac{\partial}{\partial\delta}I_{i,j}^{k+1} \cdot (\delta_{j+1}^{k+1} - \delta_j^{k+1}) \end{aligned} \tag{10.24}$$

where δ_j^{k+1} is the approach during time step $k+1$ at iteration j, which is according to (10.22) equal to

$$\delta_j^{k+1} = \max_{0\le x\le L}\left[\sum_{i=1}^m c_{i,j}^{k+1}w_i(x) - g(x - x_0 - st_{k+1})\right] - z_j^{k+1} \tag{10.25}$$

Hence, the difference $\delta_{j+1}^{k+1} - \delta_j^{k+1}$ as in (10.24) is equal to

$$\begin{aligned} \delta_{j+1}^{k+1} - \delta_j^{k+1} &= \max_{0\le x\le L}\left[\sum_{i=1}^m c_{i,j+1}^{k+1}w_i(x) - g(x - x_0 - st_{k+1})\right] \\ &- \max_{0\le x\le L}\left[\sum_{i=1}^m c_{i,j}^{k+1}w_i(x) - g(x - x_0 - st_{k+1})\right] \\ &- (z_{j+1}^{k+1} - z_j^{k+1}) \end{aligned} \tag{10.26}$$

This term cannot be simplified any further. For each j, let $x_j^{k+1} \in [0, L]$ be the value that minimizes the sum as in (10.25) during iteration j in time step k. Since x_{j+1}^{k+1} is not known beforehand, we cannot express F_{j+1}^{k+1} and $I_{i,j+1}^{k+1}$ using (10.24) explicitly. To solve this, note that the train remains at the same x position during time step j and $j+1$. Furthermore, x_{j+1}^{k+1} is independent of the height of the wheel z_{j+1}^{k+1}. We therefore make the assumption that $x_{j+1}^{k+1} \approx x_j^{k+1}$. This assumption can be justified by noting that only the coefficients $c_{i,j+1}^{k+1}$ for $1 \le i \le m$ effect the value of x_{j+1}^{k+1}. A small change of these coefficients, however, will hardly change the shape of the global deformation and hence will hardly effect the value of x_{j+1}^{k+1}.

Therefore, (10.26) can be approximated by

$$\begin{aligned}
\delta_{j+1}^{k+1} - \delta_j^{k+1} &= \sum_{i=1}^{m} c_{i,j+1}^{k+1} w_i(x_{j+1}^{k+1}) - g(x_{j+1}^{k+1} - x_0 - st_{k+1}) \\
&\quad - \sum_{i=1}^{m} c_{i,j}^{k+1} w_i(x_j^{k+1}) - g(x_j^{k+1} - x_0 - st_{k+1}) - (z_{j+1}^{k+1} - z_j^{k+1}) \\
&\approx \sum_{i=1}^{m} c_{i,j+1}^{k+1} w_i(x_j^{k+1}) - g(x_j^{k+1} - x_0 - st_{k+1}) \\
&\quad - \sum_{i=1}^{m} c_{i,j}^{k+1} w_i(x_j^{k+1}) - g(x_j^{k+1} - x_0 - st_{k+1}) - (z_{j+1}^{k+1} - z_j^{k+1}) \\
&= \sum_{i=1}^{m} (c_{i,j+1}^{k+1} - c_{i,j}^{k+1}) w_i(x_j^{k+1}) - (z_{j+1}^{k+1} - z_j^{k+1})
\end{aligned} \tag{10.27}$$

In (10.27), we eliminated the use of a maximum and hence (10.24) can be approximated by

$$\begin{aligned}
F_{j+1}^{k+1} &= F_j^{k+1} + \frac{\partial}{\partial \delta} F_j^{k+1} \cdot \left[\sum_{i=1}^{m} (c_{i,j+1}^{k+1} - c_{i,j}^{k+1}) w_i(x_j^{k+1}) - (z_{j+1}^{k+1} - z_j^{k+1}) \right] \\
I_{i,j+1}^{k+1} &= I_{i,j}^{k+1} + \frac{\partial}{\partial \delta} I_{i,j}^{k+1} \cdot \left[\sum_{i=1}^{m} (c_{i,j+1}^{k+1} - c_{i,j}^{k+1}) w_i(x_j^{k+1}) - (z_{j+1}^{k+1} - z_j^{k+1}) \right]
\end{aligned} \tag{10.28}$$

It is important to note that (10.28) is completely linear for the implicit terms at iteration $j+1$; the coefficients $w_i(x_j^{k+1})$ are known values. These expressions, combined with the approximations (10.23), can be used to linearise the equations in (10.8) with respect to the penetration. For implicit integration schemes, this yields a linear system which can then be solved for $c_{i,j+1}^{k+1}$ and z_{j+1}^{k+1}.

This Quasi-Newton approach (we only linearise with respect to a single variable using finite differences) will be applied for the Backward Euler integration scheme. This approach, however, can be applied for any other implicit scheme but this will results in even more complex expressions.

10.5.2 Applied to Backward Euler

Suppose that at each time step k and iteration j the derivatives as in (10.23) are computed. The solution of (10.11) will be computed using the just described Quasi-Newton approach. For each mode shape i, we iteratively set

$$\begin{aligned}
\dot{c}_{i,j+1}^{k+1} &= \dot{c}_i^k + \frac{\Delta t}{\rho} \left[I_{i,j+1}^{k+1} - EI\beta_i^4 c_{i,j+1}^{k+1} \right] \qquad \text{for all } 1 \leq i \leq m \\
\dot{z}_{j+1}^{k+1} &= \dot{z}^k + \Delta t \left[\frac{F_{j+1}^{k+1}}{m_c} - g \right]
\end{aligned} \tag{10.29}$$

By substituting (10.28) into (10.29), we arrive at

$$\begin{aligned}\dot{c}_{i,j+1}^{k+1} &= \dot{c}_i^k + \frac{\Delta t}{\rho}\left[I_{i,j}^{k+1} + \frac{\partial}{\partial\delta}I_{i,j}^{k+1}\cdot\left(\sum_{l=1}^{m}(c_{l,j+1}^{k+1}-c_{l,j}^{k+1})w_l(x_j^{k+1}) - (z_{j+1}^{k+1}-z_j^{k+1})\right) - EI\beta_i^4 c_{i,j+1}^{k+1}\right]\\ \dot{z}_{j+1}^{k+1} &= \dot{z}^k + \frac{\Delta t}{m_c}\left[F_j^{k+1} + \frac{\partial}{\partial\delta}F_j^{k+1}\cdot\left(\sum_{i=1}^{m}(c_{i,j+1}^{k+1}-c_{i,j}^{k+1})w_i(x_j^{k+1}) - (z_{j+1}^{k+1}-z_j^{k+1})\right) - gm_c\right]\end{aligned} \tag{10.30}$$

This can then be rearranged by moving all terms of iteration $j+1$ to the left side. We arrive at

$$\begin{aligned}&\dot{c}_{i,j+1}^{k+1} + \frac{\Delta t}{\rho}\left[EI\beta_i^4 c_{i,j+1}^{k+1} + \frac{\partial}{\partial\delta}I_{i,j}^{k+1}\cdot\left(z_{j+1}^{k+1} - \sum_{l=1}^{m}c_{l,j+1}^{k+1}w_l(x_j^{k+1})\right)\right]\\ &\qquad = \dot{c}_i^k + \frac{\Delta t}{\rho}\left[I_{i,j}^{k+1} + \frac{\partial}{\partial\delta}I_{i,j}^{k+1}\cdot\left(z_j^{k+1} - \sum_{l=1}^{m}c_{l,j}^{k+1}w_l(x_j^{k+1})\right)\right]\\ &\dot{z}_{j+1}^{k+1} + \frac{\Delta t}{m_c}\cdot\frac{\partial}{\partial\delta}F_j^{k+1}\cdot\left(z_{j+1}^{k+1} - \sum_{i=1}^{m}c_{i,j+1}^{k+1}w_i(x_j^{k+1})\right)\\ &\qquad = \dot{z}^k - \Delta t g + \frac{\Delta t}{m_c}\left[F_j^{k+1} + \frac{\partial}{\partial\delta}F_j^{k+1}\cdot\left(z_j^{k+1} - \sum_{i=1}^{m}c_{i,j}^{k+1}w_i(x_j^{k+1}\right)\right]\end{aligned} \tag{10.31}$$

Let $\mathbf{x}_j^{k+1} = [c_{1,j}^{k+1};\ldots;c_{m,j}^{k+1};z_j^{k+1}]$ and $\dot{\mathbf{x}}_j^{k+1} = [\dot{c}_{1,j}^{k+1};\ldots;\dot{c}_{m,j}^{k+1};\dot{z}_j^{k+1}]$. Then (10.31) can be written using vector notation as

$$\dot{\mathbf{x}}_{j+1}^{k+1} + \Delta t A_j^{k+1}\mathbf{x}_j^{k+1} = \dot{\mathbf{x}}^k + \Delta t\mathbf{g}_j^{k+1} \tag{10.32}$$

Here, $\mathbf{g}_j^{k+1}$ is equal to

$$\mathbf{g}_j^{k+1} = \begin{pmatrix} \frac{1}{\rho}\left[I_{1,j}^{k+1} + \frac{\partial}{\partial\delta}I_{1,j}^{k+1}\cdot\left(z_j^{k+1} - \sum_{i=1}^{m}c_{i,j}^{k+1}w_i(x_j^{k+1})\right)\right]\\ \vdots\\ \frac{1}{\rho}\left[I_{m,j}^{k+1} + \frac{\partial}{\partial\delta}I_{m,j}^{k+1}\cdot\left(z_j^{k+1} - \sum_{i=1}^{m}c_{i,j}^{k+1}w_i(x_j^{k+1})\right)\right]\\ \frac{1}{m_c}\left[F_j^{k+1} + \frac{\partial}{\partial\delta}F_j^{k+1}\cdot\left(z_j^{k+1} - \sum_{i=1}^{m}c_{i,j}^{k+1}w_i(x_j^{k+1})\right)\right] - g \end{pmatrix} \tag{10.33}$$

The matrix A changes every iteration and every time step because of the derivatives with respect to δ and the coefficients $w_i(x_j^{k+1})$. It is defined by

$$A_j^{k+1} = \begin{pmatrix} \frac{EI}{\rho}\beta_1^4 - \frac{w_1(x_j^{k+1})}{\rho}\cdot\frac{\partial}{\partial\delta}I_{1,j}^{k+1} & \cdots & -\frac{w_m(x_j^{k+1})}{\rho}\cdot\frac{\partial}{\partial\delta}I_{1,j}^{k+1} & \frac{1}{\rho}\cdot\frac{\partial}{\partial\delta}I_{1,j}^{k+1} \\ \vdots & \ddots & \vdots & \vdots \\ -\frac{w_1(x_j^{k+1})}{\rho}\cdot\frac{\partial}{\partial\delta}I_{m,j}^{k+1} & \cdots & \frac{EI}{\rho}\beta_m^4 - \frac{w_m(x_j^{k+1})}{\rho}\cdot\frac{\partial}{\partial\delta}I_{m,j}^{k+1} & \frac{1}{\rho}\cdot\frac{\partial}{\partial\delta}I_{m,j}^{k+1} \\ -\frac{w_1(x_j^{k+1})}{m_c}\cdot\frac{\partial}{\partial\delta}F_j^{k+1} & \cdots & -\frac{w_m(x_j^{k+1})}{m_c}\cdot\frac{\partial}{\partial\delta}F_j^{k+1} & \frac{1}{m_c}\cdot\frac{\partial}{\partial\delta}F_j^{k+1} \end{pmatrix} \tag{10.34}$$

Additionally, the displacement components $\mathbf{x}_{j+1}^{k+1}$ are described by the Backward Euler scheme and are equal to (see also (10.12))

$$\mathbf{x}_{j+1}^{k+1} - \Delta t\dot{\mathbf{x}}_{j+1}^{k+1} = \mathbf{x}^k \tag{10.35}$$

Now define $\mathbf{y}_j^{k+1} = [\mathbf{x}_j^{k+1}; \dot{\mathbf{x}}_j^{k+1}]$. Then (10.32) and (10.35) can be combined to get the linear system

$$B_j^{k+1}\mathbf{y}_{j+1}^{k+1} = \mathbf{y}^k + \Delta t\mathbf{h}_j^{k+1} \tag{10.36}$$

where

$$B_j^{k+1} = \begin{pmatrix} I & -\Delta tI \\ A_j^{k+1} & I \end{pmatrix}, \text{ and} \qquad \mathbf{h}_j^{k+1} = \begin{pmatrix} \mathbf{0} \\ \mathbf{g}_j^{k+1} \end{pmatrix} \tag{10.37}$$

10.5.3 Solving the linear system

At each iteration, equation (10.36) should be solved. This iteration should be repeated until convergence is reached, for example till $\|\mathbf{y}_{j+1}^{k+1} - \mathbf{y}_j^{k+1}\|/\|\mathbf{y}_j^{k+1}\| < \varepsilon$. The matrix B_j^{k+1} is a fairly small matrix so that this shouldn't be a problem. However, the matrix has an interesting structure which allows us to compute its inverse very efficiently.

Note that A_j^{k+1} can be written as as the sum of a constant diagonal matrix and a rank 1 matrix, i.e. $A_j^{k+1} = \text{diag}(\mathbf{d}) + \mathbf{e}_j^{k+1}(\mathbf{f}_j^{k+1})^T$, where

$$\mathbf{d} = \begin{pmatrix} \frac{EI}{\rho}\beta_1^4 \\ \vdots \\ \frac{EI}{\rho}\beta_m^4 \\ 0 \end{pmatrix}, \qquad \mathbf{e}_j^{k+1} = \begin{pmatrix} \frac{1}{\rho}\cdot\frac{\partial}{\partial\delta}I_{1,j}^{k+1} \\ \vdots \\ \frac{1}{\rho}\cdot\frac{\partial}{\partial\delta}I_{m,j}^{k+1} \\ \frac{1}{m_c}\cdot\frac{\partial}{\partial\delta}F_j^{k+1} \end{pmatrix}, \text{ and} \qquad \mathbf{f}_j^{k+1} = \begin{pmatrix} -w_1(x_j^{k+1}) \\ \vdots \\ -w_m(x_j^{k+1}) \\ 1 \end{pmatrix} \tag{10.38}$$

The following Lemmas will show that because of the special structure of A_j^{k+1}, equation (10.36) can be solved explicitly.

Lemma 10.1. The inverse of B_j^{k+1} is given by

$$(B_j^{k+1})^{-1} = \begin{pmatrix} (X_j^{k+1})^{-1} & \Delta t(X_j^{k+1})^{-1} \\ \frac{1}{\Delta t}((X_j^{k+1})^{-1} - I) & (X_j^{k+1})^{-1} \end{pmatrix}, \text{ where } \qquad X_j^{k+1} = I + \Delta t A_j^{k+1} \tag{10.39}$$

Proof: A simple multiplication yields

$$\begin{aligned} B_j^{k+1} & \begin{pmatrix} (X_j^{k+1})^{-1} & \Delta t(X_j^{k+1})^{-1} \\ \frac{1}{\Delta t}((X_j^{k+1})^{-1} - I) & (X_j^{k+1})^{-1} \end{pmatrix} \\ &= \begin{pmatrix} I(X_j^{k+1})^{-1} - \Delta t I \frac{1}{\Delta t}((X_j^{k+1})^{-1} - I) & I\Delta t(X_j^{k+1})^{-1} - \Delta t I(X_j^{k+1})^{-1} \\ A_j^{k+1}(X_j^{k+1})^{-1} + I\frac{1}{\Delta t}((X_j^{k+1})^{-1} - I) & A_j^{k+1}\Delta t(X_j^{k+1})^{-1} + I(X_j^{k+1})^{-1} \end{pmatrix} \\ &= \begin{pmatrix} I & 0 \\ \frac{1}{\Delta t}((\Delta t A_j^{k+1} + I)(X_j^{k+1})^{-1} - I) & (I + \Delta t A_j^{k+1})(X_j^{k+1})^{-1} \end{pmatrix} \\ &= I \end{aligned} \tag{10.40}$$

which proves the Lemma. □

Therefore, our goal is to compute the inverse of X_j^{k+1}. The following Lemma gives an explicit expression for this inverse

Lemma 10.2. The inverse of X_j^{k+1} is given by

$$(X_j^{k+1})^{-1} = D - \frac{\Delta t D \mathbf{e}_j^{k+1}(\mathbf{f}_j^{k+1})^T D}{1 + \Delta t(\mathbf{f}_j^{k+1})^T D \mathbf{e}_j^{k+1}} \tag{10.41}$$

where D is the diagonal matrix given by $D_{ii} = (1 + \Delta t d_i)^{-1}$.

Proof:

First, note that

$$X_j^{k+1} = I + \Delta t A_j^{k+1} = (I + \Delta t \text{diag}(\mathbf{d})) + (\Delta t \mathbf{e}_j^{k+1})(\mathbf{f}_j^{k+1})^T \tag{10.42}$$

Hence X_j^{k+1} is, similarly to A_j^{k+1}, a rank-1 update of a diagonal matrix. According to the Sherman-Morrison formula (see [19]), the inverse is given by

$$(X_j^{k+1})^{-1} = (I + \Delta t \text{diag}(\mathbf{d}))^{-1} - \frac{(I + \Delta t \text{diag}(\mathbf{d}))^{-1}(\Delta t \mathbf{e}_j^{k+1})(\mathbf{f}_j^{k+1})^T(I + \Delta t \text{diag}(\mathbf{d}))^{-1}}{1 + (\mathbf{f}_j^{k+1})^T(I + \Delta t \text{diag}(\mathbf{d}))^{-1}(\Delta t \mathbf{e}_j^{k+1})} \tag{10.43}$$

By rearranging the terms and noting that $I + \Delta t \text{diag}(\mathbf{d})$ is a diagonal matrix and thus its inverse is the diagonal matrix with the inverted components, we arrive at the result. □

Lemma 10.1 and 10.2 can be combined to get an explicit expression for the solution of (10.36) at each iteration. At each iteration, we set $\mathbf{y}_{j+1}^{k+1} = (B_j^{k+1})^{-1}[\mathbf{y}^k + \Delta t \mathbf{h}_j^{k+1}]$ using the inverse of B_j^{k+1} as in Lemma 10.1. Although this matrix changes every iteration and time step, the expression for $\mathbf{y}_{j+1}^{k+1}$ is explicit. No inverse needs to be computed, we only need to compute D once which is extremely easy since it is a diagonal matrix.

10.6 The algorithm

Having discussed the Quasi-Newton approach, we will combine all theory discussed in this report to construct an algorithm to simulate train-bridge contact. The algorithm is applied for the Backward Euler integration scheme, but other (implicit) time integration schemes can be applied as well. The only difference here is that the expression (10.36) is no longer valid; the integration scheme should be applied to get a similar expression.

Compared with the algorithm using Picard iteration, the Quasi-Newton approach requires a bit more computation per iteration. At each iteration, CONTACT needs to be run twice instead of once; one for a given penetration $\delta(x)$ and one for the penetration $\delta(x) + \alpha$ for α for small value. This is used to compute the derivatives as in (10.23). Additionally, the value x_j^{k+1}, i.e. the x position of the initial point of contact at iteration j, should be computed. This value is required to compute the matrix B_j^{k+1}. Solving (10.36) can either be done by constructing B_j^{k+1} and solving the linear system, or by computing $(B_j^{k+1})^{-1}[\mathbf{y}^k + \Delta t \mathbf{h}_j^{k+1}]$ directly by using Lemma 10.1 and 10.2.

The Quasi-Newton approach is therefore per iteration approximately twice as computationally expensive as the Picard approach. However, it is far more stable and converges much quicker. Numerical results show that under relaxation is no longer needed to achieve converge. It can still be used similarly as in (10.17) and can be advantageous if the under relaxation factor ω is taken small. This can be used to prevent additional overshoot during each iteration. In practice it appeared that the under relaxation is mostly useful during the first few iterations. Hence, the under relaxation factor ω_j can be a function of j. For the best convergence speed, ω_j should be near 1 (i.e. only a small amount of under relaxation is used) for larger j.

The full algorithm is described in Algorithm 10.2. For the complete (more detailed) Matlab implementation, see Appendix A.5.

Algorithm 10.2 Performing a full train-bridge simulation using the Quasi-Newton approach

Set $c_i^0 = \dot{c}_i^0 = 0$ for all $1 \le i \le m$. If the initial condition is not zero, use (8.61) and (8.62) instead.
Set $z^0 = \dot{z}^0 = 0$ or equal to the corresponding initial conditions.
Set $u^0(x) \equiv \sum_{i=1}^m c_i^0 w_i(x)$.
for $k = 1$ to n **do**
    Set $c_i = c_i^{k-1}$ for all $1 \le i \le m$.
    Set $z = z^{k-1}$.
    repeat
        Set $c_i^* = c_i$ for all $1 \le i \le m$.
        Set $z^* = z$.
        Set $\delta(x) = \sum_{i=1}^m c_i^* w_i(x) - z^* - g(x - x_0 - st^k)$ for all x.
        Approximate xpos = $\operatorname{argmax}_{x\in[0,L]}\delta(x)$.
        Compute $p(x)$, F and $l(x)$ with CONTACT using penetration $\delta(x)$.
        Compute $p_2(x)$ and F_2 with CONTACT using penetration $\delta(x) + \alpha$.
        Set $\dot{F} = (F_2 - F)/\alpha$.
        for $i = 1$ to m **do**
            Approximate int = $\int_0^L p(x)w_i(x)$ using a Newton-Cotes formula.
            Approximate $\text{int}_2 = \int_0^L p_2(x)w_i(x)$ using a Newton-Cotes formula.
            Set $\dot{I}_i = (\text{int}_2 - \text{int})/\alpha$.
        end for
        Compute **h** as in (10.33) and (10.37).
        Solve (10.36) for $\mathbf{y} = [\mathbf{c}; z; \dot{\mathbf{c}}; \dot{z}]$ (can be done explicitly with Lemma 10.1 and 10.2)
        Set error = $\|[\mathbf{c} - \mathbf{c}^*; z - z^*]\|_2$.
    until error $\le \varepsilon$
    Set $u^k(x) = \sum_{i=1}^m c_i w_i(x)$ for all x
    Set $u_{\text{tot}}^k(x) = u^k(x) + \frac{1}{2}l(x)$ for all x.
    Set $w^k(x) = g(x - x_0 - st^k) + z$ for all x.
    Set $w_{\text{tot}}^k(x) = w^k(x) - \frac{1}{2}l(x)$ for all x.
end for

10.7 Numerical results

Using the properties as given in Table 10.1, the Quasi-Newton approach has been applied to perform a full train-bridge simulation for the Backward Euler integration scheme. The algorithm converges, and each time step only takes an average of 3 iterations before convergence has been reached.

10.7.1 Visual observations

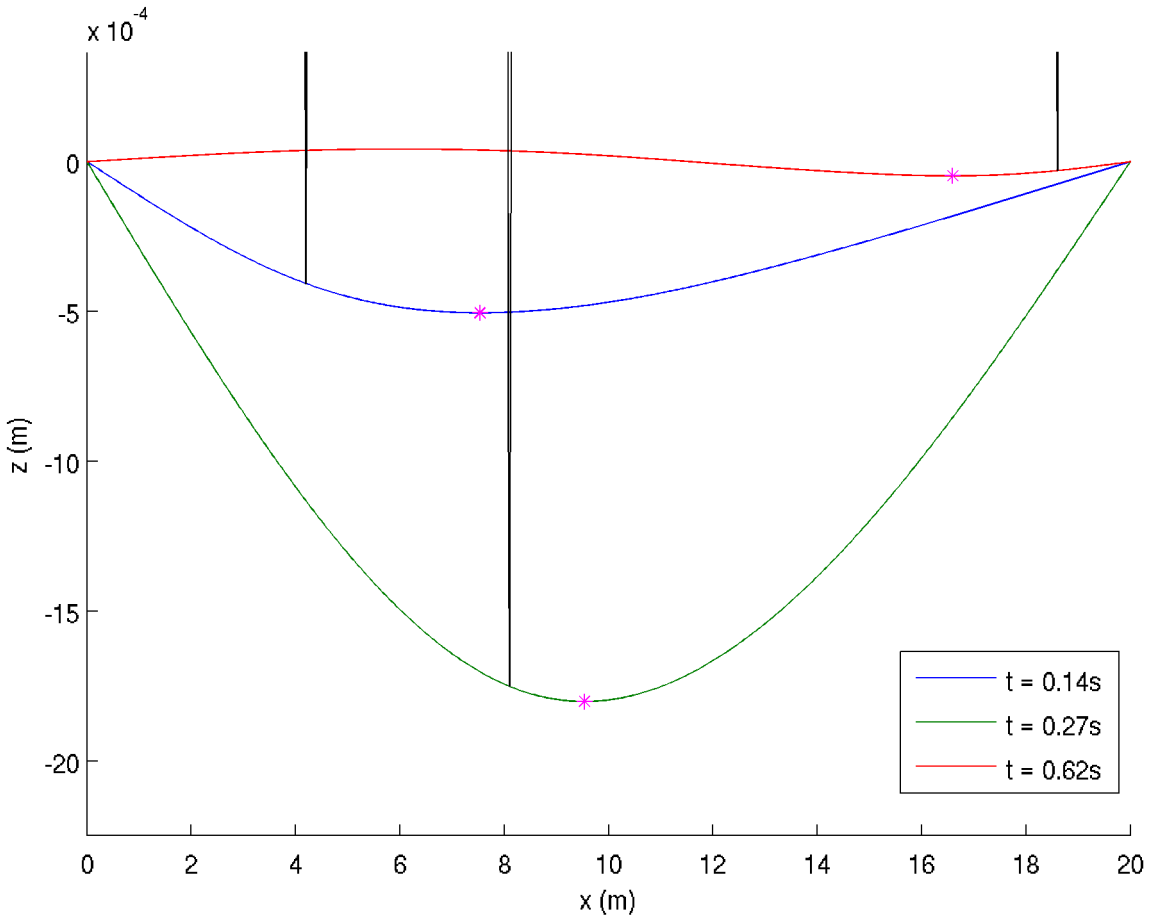


Figure 10.1: Representation of the deformation of the deformed bridge at 3 different time steps. The black lines represent the wheel at the corresponding time. The pink stars represent the lowest points of the bridge.

In Figure 10.1, the deformation of the bridge is shown for 3 different time steps. Note that the local deformation of both the bridge and the wheel is very small with respect to the global deformation. As expected, the bridge slowly deforms around the left side. This asymmetry becomes less visible as the train moves further. Interesting is that at the moment right before

the train has passed the bridge, the bridge becomes S-shaped as the second mode shape becomes dominant.

One aspect we are interested in is the influence of the speed of the train on the deformation of the bridge. Consider the exact same problem, but now suppose that $s = 6$m/s, i.e. 5 times as slow as the previous example. The results can be shown in Figure 10.2.

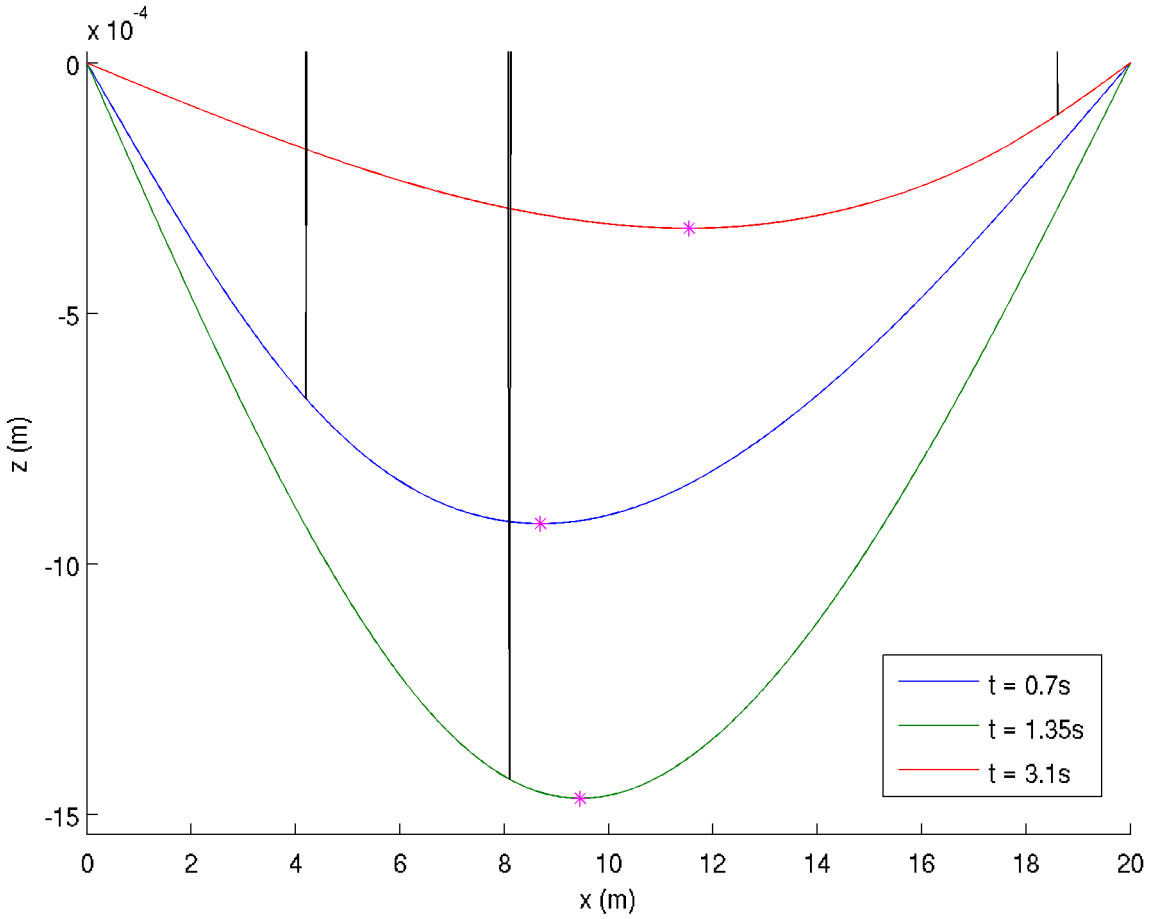


Figure 10.2: Representation of the deformation of the deformed bridge at 3 different time steps. Here, the speed of the train is only 6m/s.

As one can see, the results are similar. The bridge starts to deform at its left side, and the asymmetry continues to move to the right. The maximum deformation of approximately 1.5mm is slightly lower but remains of the same magnitude as for the case when $s = 30$m/s.

An interesting difference is that there is less vibration near the moment the train has passed the end of the bridge. This can be explained by the fact that around this time, the gravitational force being exerted on the bridge will cause additional damping. This is because the train moves five times as slow, and hence the damping caused by gravity remains five times as long.

In Figure 10.3 the first four modal coefficients are plotted as function of the time. As expected,

the first modal coefficient has by far the largest amplitude. Interesting is that small oscillations occur. These oscillations, however, become smaller as time passes. This is most likely the result of the numerical damping of the Backward Euler scheme. For a better approximation, another numerical scheme with better energy conservation properties such as the Newton-method should be applied instead.

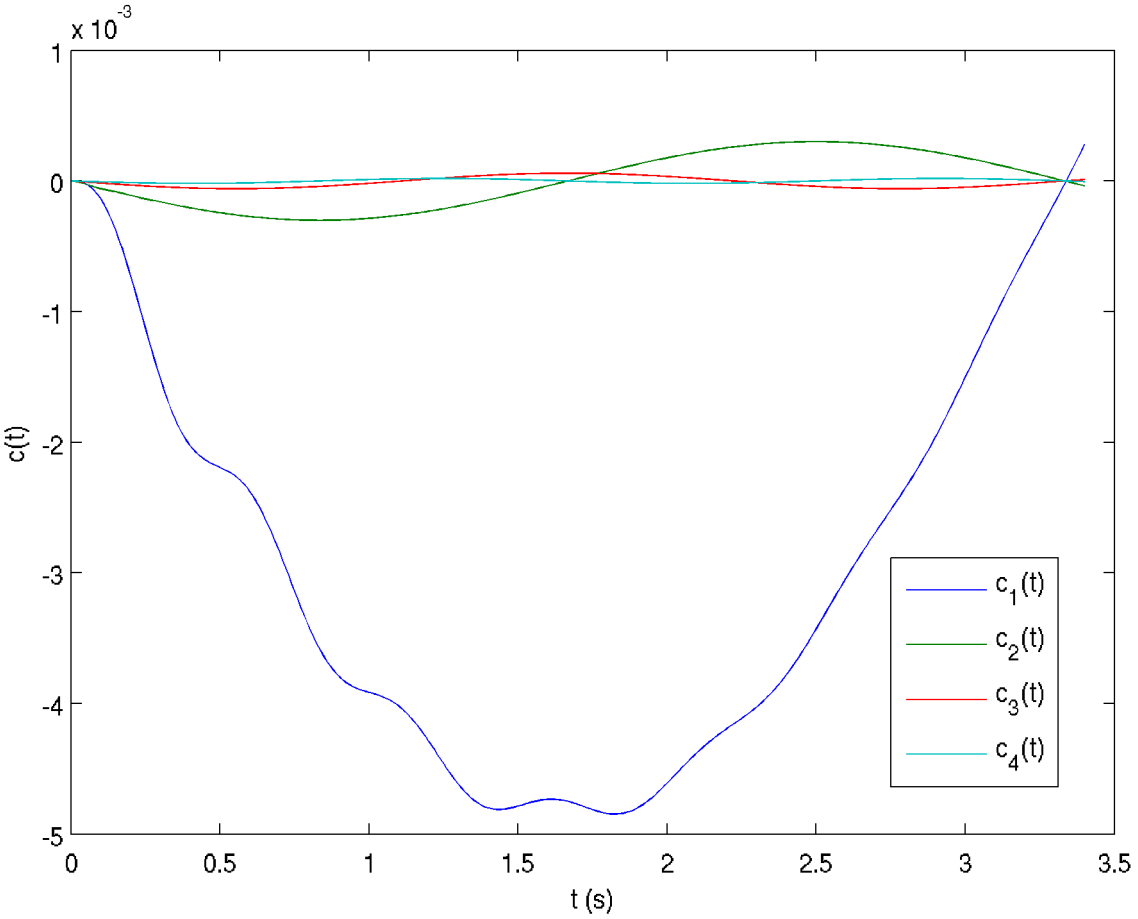


Figure 10.3: The first four modal coefficients $c_i(t)$.

10.7.2 Effect on the approach and dynamic amplification factor

In Figure 10.4, the approach $\delta(t)$ between the train wheel and bridge can be seen. Throughout the simulation, the approach hovers around $\delta(t) \approx 4.64 \cdot 10^{-7}$, except near $t = 0$, where $\delta(0) = 0$. As we can see, the approach tends to be larger in the middle of the simulation. An explanation for this behaviour is that around this time, the global deformation was of the largest magnitude. It appears that the global deformation does indeed have a reasonable large impact on the local deformation.

This observation can also be made by applying Hertz theory. Near the contact area, the surface of the bridge can be approximated by a cylinder of a large radius R_2. If the bridge is

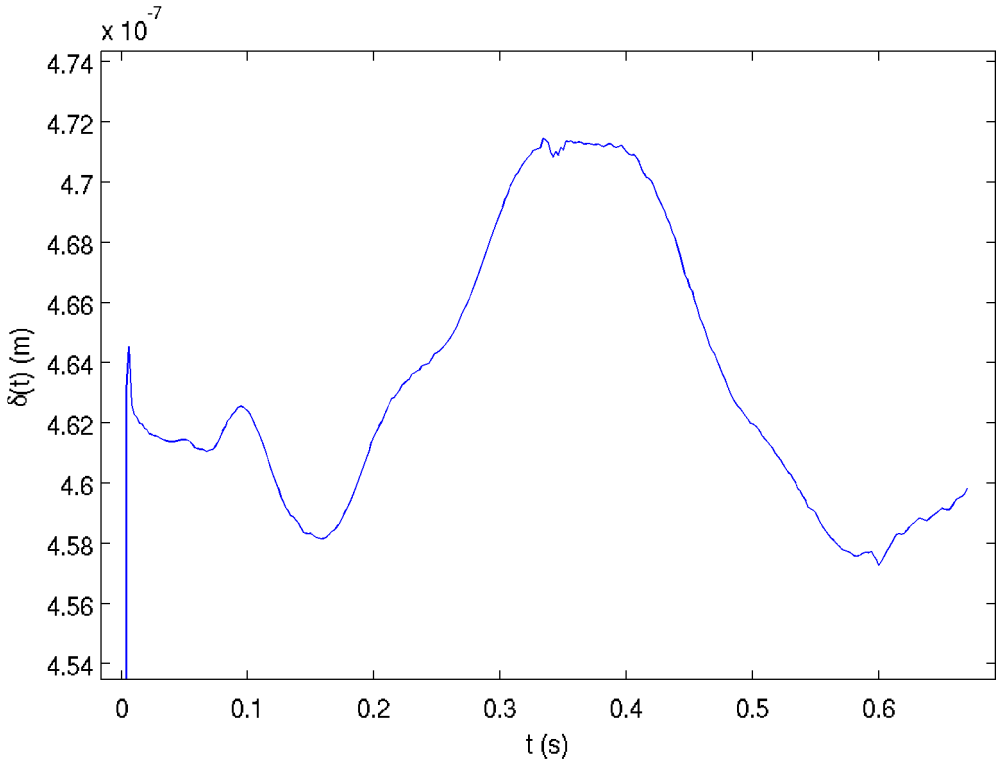


Figure 10.4: The approach between the train wheel and the bridge as function of time.

globally undeformed, then we say that this radius is $R_2 = \infty$. The wheel can be represented by a cylinder of radius R_1. From (4.17) and (4.21) it follows that

$$\delta \approx \frac{a^2}{R} = a^2 \left(\frac{1}{R_1} + \frac{1}{R_2} \right) \tag{10.44}$$

As the global deformation becomes larger, the radius R_2 decreases. From (10.44) it follows that this will result in a larger approach, exactly what we have noticed in Figure 10.4.

Additionally, Figure 10.4 shows local oscillations of different magnitude and seem fairly random. These oscillations occur due to the inertia of the wheel, similar as the result we have seen in Chapter 6.

The dynamic amplification factor is the relative difference between the gravitational force and the force exerted by the bridge. This is therefore equal to $F_n(t)/(m_c g)$. This factor depends on the time; a plot can be seen in Figure 10.5.

The dynamical amplification factor is directly related to the normal force and thus also to the approach δ, hence the figure is similar as in Figure 10.44. As we can see, the dynamical amplification factor hovers around 1, i.e. the normal force exerted by the bridge is roughly equal to the gravitational force. The maximum amplification and dissipation factor are approximately 1.8% and 1.3%, respectively.

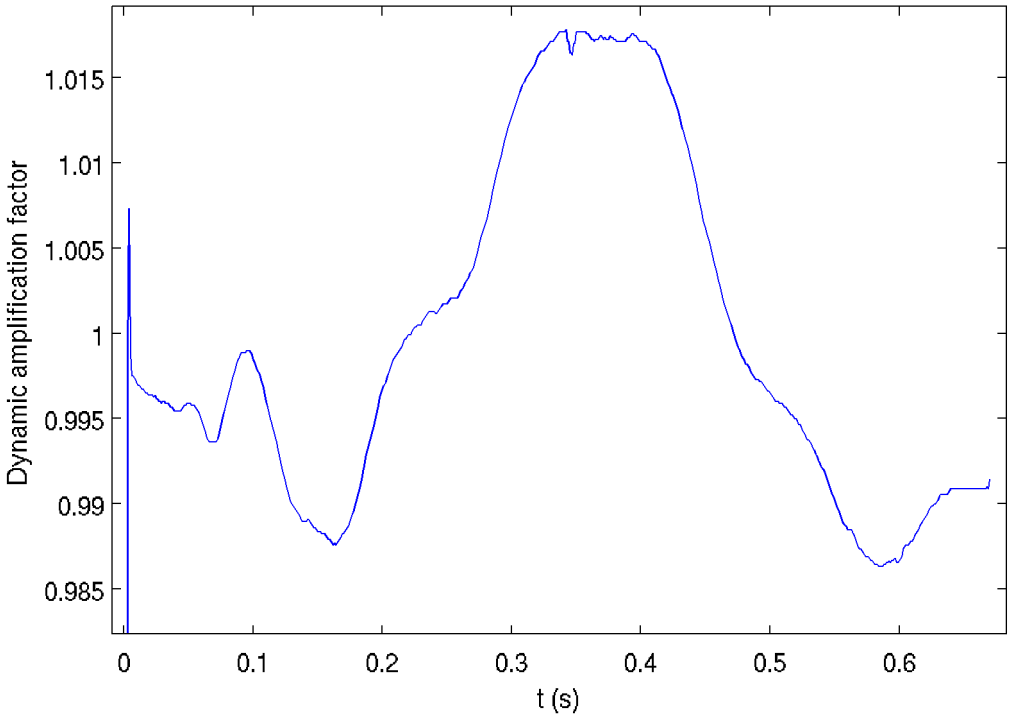


Figure 10.5: The dynamic amplification factor as function of time.

10.7.3 Numerical convergence

In Theorem 8.3, we have seen that the error of the beam equation by using m modes is $\|u - u_m\|_2 = \mathcal{O}(m^{-3})$. However, we have only proved this for the stationary case. We will show from the numerical results that this convergence rate remains valid for the instationary case. In order to do so, we have computed the norm $\|c_i\|_\infty = \max_{t\in[0,t_0]}|c_i(t)|$ for a number of modal coefficients c_i. The results can be seen in Table 10.3.

i	$\|c_i\|_\infty$	$\|c_i - c_{2i}\|_\infty / \|c_{2i} - c_{4i}\|_\infty$
1	$7.3 \cdot 10^{-3}$	24.231
2	$3.09 \cdot 10^{-4}$	16.116
4	$1.92 \cdot 10^{-5}$	16.048
8	$1.20 \cdot 10^{-6}$	16.063
16	$7.45 \cdot 10^{-8}$	15.998
32	$4.67 \cdot 10^{-9}$	16.004
64	$2.90 \cdot 10^{-10}$	16.055

Table 10.3: The sup norm of each modal coefficient.

By Richardson extrapolation, we can show that the norm $\|c_m\|_\infty = \mathcal{O}(m^{-4})$. Using Taylor expansion, we write $\|c_m\|_\infty = K(1/m)^p + \mathcal{O}((1/m)^{p+1})$ some $K, p > 0$. Note that

$$\begin{aligned}\|c_m\|_\infty - \|c_{2m}\|_\infty &= K((1/m)^p - (1/(2m))^p) = K(1/m)^p(1-(1/2)^p) + \mathcal{O}((1/m)^{p+1}) \\ \|c_{2m}\|_\infty - \|c_{4m}\|_\infty &= K((1/(2m))^p - (1/(4m))^p) = K(1/m)^p(1/2)^p(1-(1/2)^p) + \mathcal{O}((1/m)^{p+1})\end{aligned} \tag{10.45}$$

By dividing the second from the first equation, we arrive at

$$\frac{\|c_m\|_\infty - \|c_{2m}\|_\infty}{\|c_{2m}\|_\infty - \|c_{4m}\|_\infty} = 2^p + \mathcal{O}((1/m)^{p+1}) \tag{10.46}$$

This relative difference in norm is also shown in Table 10.3. It seems clear that $\|c_i - c_{2i}\|_\infty / \|c_{2i} - c_{4i}\|_\infty \to 16$ as i becomes larger. Hence, by (10.46), it follows that $2^p = 16$, i.e. $p = 4$. In conclusion, $\|c_m\|_\infty = \mathcal{O}(m^{-4})$ seems to hold.

Note that this conclusion is in agreement with Theorem 8.3. The error $u - u_m$ consists of the terms c_i for $i = m+1$ to infinity. By summing of each of these terms, we lose an order of convergence, i.e. $\|u - u_m\|_\infty = \mathcal{O}(m^{-3})$.

Chapter 11

Summary and Further Research

11.1 Summary and conclusions

In this thesis, we have discussed the basics of non frictional contact problems. The main goal of this thesis was to figure out how CONTACT can be used for dynamical contact problems involving two objects interacting with each other. Specifically, we were looking at a contact problems occurring in the train industry; how does a (for instance) bridge deform when a train is moving over it and how can this deformation be computed efficiently?

We have argued that two different phenomena occur in this problem, namely global and local deformations. Local deformation occurs solely around the initial point of contact and involves the material of the object being compressed. This compression results in a force pointing in the normal direction. Such a problem has been discussed in Chapter 6. Here, we have looked at the problem involving a rigid sphere (or any other arbitrary object) dropping on an elastic half plane. A differential equation for the height of the sphere has been derived by using Newton's second law of motion by computing the normal force exerted by the half space using Hertz theory or by running CONTACT.

By using a time integration scheme for this simple one-dimensional differential equation, we were capable of computing the approach of the contact between the sphere and half-space as function of time. Multiple time-integration schemes have been discussed. We have argued that implicit integration schemes are the best choices for this problem, especially for stiff materials such as steel. Radau5 (or the Radau family of integrations schemes as a whole) turned out to be effective due to their unconditional stability and their high (adaptable) order of accuracy. The computationally inexpensive Verlet and the Leapfrog method are also very useful due to their energy conservation. For very stiff problems, the Newmark-beta method is the most practical scheme. This is because Newmark-beta is both energy conserving as well as stable because it is an implicit scheme.

CONTACT can also be used for this problem. Not only will this result in the same contact force as in Hertz theory, CONTACT will also result the quasi-static deformation of the half-plane. This spatial deformation is the solution of the quasi-static elasticity equation for the particles at the surface of the half plane. This can be used to approximate the local deformation for our problem. However, this deformation is not valid for dynamical contact problems; it is based on the assumption that inertia is neglected. Hence, the quasi-static deformation occurs instantaneously, ignoring shock waves occurring right after the moment

of impact. In reality, inertia plays a role in dynamical contact problems and should also be taken into account.

This can be done by discretising the linear elasticity equations. These equations form a so-called differential algebraic equation. We have shown in Chapter 7 how this differential equation can be discretised using a finite difference approach and how this can be solved with an integration scheme and at the same time be combined with the algebraic relations between the elasticity components. Using this expression, the same contact problem as discussed before has been solved. The result was fairly similar to the quasi-static result computed using CONTACT, but the main differences occur directly after the moment of impact and after the moment contact was lost. In these moments, the waves resulting from the inertia of the elasticity equations were most visible and not existing in the quasi-static solution.

In this thesis, we have proposed a different, much less computational expensive, way of computing the deformation of a bridge. We have modelled the deformation of the bridge as the sum of the global and local deformation. The local deformation is approximated using the quasi-static results computed by CONTACT. The theory behind global deformations has been thoroughly discussed in Chapter 8. By using mode shapes to approximate the solution of the beam equation, we have arrived at a system of independent ordinary differential equations. These equations can be solved by using a time integration scheme combined with a Newton-Cotes formula. This approach has several advantages; the main one being that the differential equations for the modal coefficients are independent of each other. Furthermore, the solution converges quickly to the analytical solution so that only a few (usually 50 at most) mode shapes are needed. It also has favourable stability properties. Additionally, the usual finite difference approach will only approximate the deformation at given grid points. This is not the case for the modal approach, which results in a C^∞ function which can be evaluated at arbitrary $x \in [0, L]$ without the need to interpolate. This is especially useful to compute modal response integrals for pressure distributions which have a very small support, which is the case for the train-bridge contact problem.

For this contact problem, the main problem that occurs is that the global and local deformation of the bridge are not independent of each other. The external pressure used by the beam equation is computed using CONTACT, which requires the global deformation to be known. To compute the stationary solution, we have proposed a simple algorithm that combined the two in Chapter 9. For the instationary problem, this turned out to be much harder. The algorithm turned out to break down very easily for implicit time integration schemes.

Hence, we have proposed a Newton-Raphson like method. We have argued that linearising with respect to the approach is the most computationally efficient way of achieving stability. Additionally, the linearisation allows the iterative process to converge much faster compared to the usual Picard iteration, usually only needing three iterations per time step. The resulting expression for implicit time integration schemes is a linear system. Originally, it seemed that this linear system was dense; so that the computational power required to solve this system would be significant. However, we have shown that this (dense) matrix corresponding to this system contains some very useful properties; it can be written as the sum of a constant diagonal matrix and a rank-one matrix. This makes it possible to solve the linear system explicitly. This unique property is the result of the fact that the differential equations for the mode shapes are independent.

Overall, the modal approach combined with the quasi-static local deformation is a very effective way of computing the total deformation of a bridge. Although inertia of the particles around the contact area is ignored, the beam equation will reintroduce inertia as a global phenomenon. The local and global deformation have been successfully combined by

using CONTACT. Combined with an implicit time integration scheme such as Radau5 or Newmark-beta, the resulting algorithm is an accurate, but most importantly, extremely computationally efficient way of of computing the total deformation of a bridge compared with the usual finite difference (or finite element) approach.

11.2 Further research

As an interesting extension to the problem, one could take tangential forces (friction) into account. Often, the circumferential velocity $2\pi\omega R$ of a train wheel is not exactly equal to the overall forward velocity V. If the train is accelerating, for example, the circumferential speed of particles at the wheel boundary is larger than overall velocity of the train. This creates creepage between the wheel and rail in the tangential direction. The contact area of the wheel will be split into two different parts; the adhesion and the slip area, as can be seen in Figure 11.1. The physics of this contact is described by Kalker's rolling contact model [20].

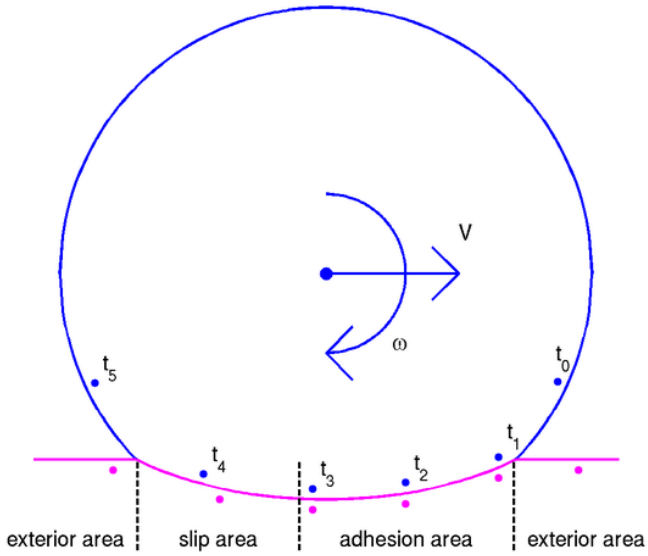


Figure 11.1: The different contact areas of the wheel that occur during rolling. Source: [5].

CONTACT has implemented this model and is capable of taking creepage into account for the contact model. Further research should therefore be done to figure out how this can be used for dynamical contact problems.

Bibliography

[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] Vollebregt, E.A.H. (2013). User guide for CONTACT, Vollebregt & Kalker's rolling and sliding contact model. TR09-03, version 13.1.

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

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

[8] Toxvaerd, S., Heilmann, O. J., & Dyre, J. C. (2012). Energy conservation in molecular dynamics simulations of classical systems. The Journal of chemical physics, 136(22), 224106.

[9] Skeel, R. D. (1993). Variable step size destabilizes the Störmer/leapfrog/Verlet method. BIT Numerical Mathematics, 33(1), 172-175.

[10] Newmark, N. M. (1959). A method of computation for structural dynamics. Journal of the Engineering Mechanics Division, 85(3), 67-94.

[11] 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.

[12] 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.

[13] Butcher, J. C. (1987). The numerical analysis of ordinary differential equations: Runge-Kutta and general linear methods. Wiley-Interscience.

[14] Sli, E., & Mayers, D. F. (2003). An introduction to numerical analysis. Cambridge university press.

[15] Timoshenko, S. (1953). History of strength of materials: with a brief account of the history of theory of elasticity and theory of structures. Courier Corporation.

[16] Vuik, C., van Beek, P., Vermolen, F., & van Kan, J. (2006). Numerieke methoden voor differentiaalvergelijkingen (Vol. 159). VSSD.

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

[18] Fornberg, B. (1988). Generation of finite difference formulas on arbitrarily spaced grids. Mathematics of computation, 51(184), 699-706.

[19] Sherman, J., & Morrison, W. J. (1950). Adjustment of an inverse matrix corresponding to a change in one element of a given matrix. The Annals of Mathematical Statistics, 124-127.

[20] Vollebregt, E. A. H. (2009). Refinement of Kalkers rolling contact model. In In: Proceedings of the 8th international conference on contact mechanics and wear of rail/wheel systems, Firenze (pp. 149-156).

Appendix A

Matlab codes

A large amount Matlab codes have been written and used for this report. The most important ones are shown here.

A.1 run_contact.m

The following code provides a functionality to communicate between Matlab and CONTACT.

```
1  function [Fc,pressure,deformed] = run_contact(penetration, settings)
2  % Computes the normal force & pressure as well as the quasi-static
3  % deformed distance of the surface.
4  % The user should supply the penetration in a rectangular uniform grid
5  % as well as various other variables such as the elasticity coefficients.
6  % This should be supplied using a Matlab struct.
7
8  if min(penetration(:)) >= 0
9      % No penetration, we do not need to run CONTACT.
10     % All output is zero.
11     Fc = 0;
12     [deformed,pressure] = deal(zeros(size(penetration)));
13 else
14     % There is penetration.
15
16     % Creating the input file:
17     fid = fopen('temp.inp', 'w');
18     fprintf(fid,' 3 module %% result element 1, Contact patch 1\n');
19     fprintf(fid,'%% Next case 1\n');
20     fprintf(fid,' 200020 P-B-T-N-F-S PVTIME, BOUND , TANG , NORM , FORCE , STRESS\n')
           ;
21     fprintf(fid,' 022020 L-D-C-M-Z-E FRCLAW, DISCNS, INFLCF, MATER , RZNORM, EXRHS\
           n');
22     fprintf(fid,' 002111 G-I-A-O-W-R GAUSEI, IESTIM, MATFIL, OUTPUT, FLOW , RETURN\n
           ');
23     fprintf(fid,' 200 30 30 1 1.0E-04 MAXGS , MAXIN , MAXNR , MAXOUT, EPS\n');
24     fprintf(fid,' 0.000 0.000 0.000 0.000 FUN, FUX, FUY, CPHI\n');
```

```
25      fprintf(fid,'%% Note: N=1 means FUN == FN, F=0 means FUX == CKSI, FUY == CETA\n');
26      fprintf(fid,' 0.3000 0.3000 FSTAT, FKIN\n');
27      fprintf(fid,sprintf(' %.4f %.4f %.4E %.4E SIGMA 1,2, GG 1,2\n',settings.nu,settings.nu,settings.G,1
            E+20));
28      fprintf(fid,' 1 IPOTCN\n');
29      fprintf(fid,sprintf(' %3d %3d %6.3f %6.2f %5.8f %5.8f MX,MY,XL,YL,DX,DY\n',size(penetration
            ,2), size(penetration,1), -size(penetration,2)/2 * settings.dx, -size(penetration,1)/2 *
            settings.dy, settings.dx, settings.dy));
30      fprintf(fid,' 9 1 IBASE, IPLAN\n');
31      fprintf(fid,'%% PENETRATION, (1)-(2): SPECIFIED PER ELEMENT');
32
33      for i = 1:size(penetration,1)
34          fprintf(fid,'\n');
35          for j = 1:ceil(size(penetration,2) / 5)
36              fprintf(fid,'\n ');
37              p = penetration(i,5*(j-1)+1:min(size(penetration,2),5*(j-1)+5));
38              fprintf(fid,sprintf(' %.4E',p));
39          end
40      end
41
42      fprintf(fid,'\n%% UNRESTRICTED PLANFORM');
43      fprintf(fid,'\n 0 module');
44      fclose(fid);
45
46      % Running CONTACT:
47      tic;
48      system('../i01/contact_v13.1/bin/contact 2 temp.inp');
49      disp(['Running CONTACT took ',num2str(toc),' seconds.']);
50
51      % Reading the contact force
52      fid = fopen('temp.out');
53
54      while isempty(findstr(fgets(fid), 'TOTAL FORCES'))
55      end
56
57      fgets(fid);
58      array = str2num(fgets(fid));
59      Fc = array(1);
60      Fx = array(2);
61      Fy = array(3);
62
63      % Reading the deformed distance and pressure
64      data = load('temp.0001.mat','-ascii');
65      data = data(4:end,:);
66
67      deformed = reshape(data(:,8), size(penetration'))';
68      pressure = reshape(data(:,5), size(penetration'))';
69  end
70  end
```

A.2 SphereOnPlaneCONTACT.m

This Matlab code computes the height of the sphere as discussed in Chapter 6. CONTACT is used to determine the normal force F_n as well as the quasi-static deformation of the half-plane. Differential equation (6.3) is integrated using some of the integration schemes of Chapter 5, specifically Radau5, Runge-Kutta 4 and Verlet.

```
% Computes the rigid height of a rigid sphere falling on an
% elastic half-plane. Verlet, Radau and RK schemes are implemented.
% Plots the height as function of the time as well as the
% quasi-static deformation of the half-plane.

% Global settings:
settings.g = 9.81; % Gravitational acceleration
R = 0.10; % Radius of ball
z0 = 1; % Height of ball at t=t0
v0 = 0; % Speed of ball at t=t0

% 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:
eps = 0.0001; % Max picard error
dt = .001;
t0 = 0;
t1 = 1;

% End of configuration

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

% Creating the ball
settings.dx = .01;
settings.dy = .01;
x = -R/2:settings.dx:R/2;
y = -R/2:settings.dy:R/2;
[X,Y] = meshgrid(x,y);
Z = R - sqrt(R^2 - X.^2 - Y.^2);
Z(imag(Z) ~= 0) = R;
deformed = zeros([size(Z), n]);
Fg = m * settings.g;
Fn = zeros(n,1);

% We are solving the differential equation z'' = A(z), where
A = @(F) F / m - settings.g;
```

```
48
49
50  if 1
51  % Verlet
52  for k = 2:n
53      [Fn(k),pressure,deformed(:,:,k)] = run_contact(Z + z(k - 1), settings);
54
55      Fres = Fn(k) - Fg;
56      a(k) = Fres / m;
57
58      % Integrating using Verlet
59      if k == 2
60          z(2) = z(1) + v0 * dt + 1/2 * a(1) * dt^2;
61      else
62          z(k) = 2 * z(k - 1) - z(k - 2) + a(k - 1) * dt^2;
63      end
64  end
65  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(run_contact(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              % Computing deformation
100             [Fn(k),pressure,deformed(:,:,k)] = run_contact(Z + z(k), settings);
101             break;
102         end
```

```
103     end
104 end
105 end
106
107
108 if 0
109 % RK4
110 % Butcher tableau:
111     % IIA Method
112     T = [0,0,0,0;
113          1/2,0,0,0;
114          0,1/2,0,0;
115          0,0,1,0];
116     b = [1/6, 1/3, 1/3, 1/6];
117     [kz,kv] = deal(zeros(4,1));
118 for k = 2:n
119     kzo = kz;
120     kvo = kv;
121     for i = 1:4
122         kz(i) = v(k - 1) + dt * T(i,:) * kvo;
123         kv(i) = A(run_contact(Z + z(k - 1) + dt * T(i,:) * kzo, settings));
124     end
125     z(k) = z(k - 1) + dt * b * kz;
126     v(k) = v(k - 1) + dt * b * kv;
127     [Fn(k),pressure,deformed(:,:,k)] = run_contact(Z + z(k), settings);
128 end
129 end
130
131
132 % Plotting the quasi−static deformation
133 if 1
134 f = figure;
135 xlabel('x (m)');
136 ylabel('y (m)');
137 zlabel('z (m)');
138 for k = 1:n
139     if z(k) < .2
140         surf(X, Y, Z + z(k)); % The ball (bottom)
141         hold on;
142         surf(X, Y, −Z + 2∗R + z(k)); % The ball (top)
143         surf(X, Y, −deformed(:,:,k)) % The surface
144         axis equal;
145         axis([−R, R, −R, R, min(z), .1 + 2 * R]);
146         hold off;
147         pause(dt);
148     end
149 end
150 end
```

A.3 ElasitictySphereHalfplane.m

In this code, the contact between a rigid cylindrical object and an elastic-half plane is simulated. The linear elasticity equations have been discretised and solved using Backward Euler such as described in Chapter 7.

```
% Discretisation of the half-plane
% Based on the equations:
% nabla * sigma + F = rho u''
% eps = 1/2 [nabla u + (nabla u)^T]
% sigma = lambda Tr(eps)I + 2Ge

if 1
% Configuration here:
mx = 31;
my = 15;
hx = .3;
hy = .1;

dtbig = 0.001; % If there is penetration, using dtsmall
dtsmall = 0.0001; % instead of dtbig
dtplot = 0.001;
t1 = .5;

% General settings:
E = 200E+3; % steel = 200GPa
nu = .285; % steel = .285;
rho = 7900; % steel = 7900kg/m^3
g = 9.81;

R = 0.1; % Radius of cylinder
L = 1; % Length of cylinder
z0cyl = .1; % Height of cylinder at t=t0
v0cyl = 0; % Speed of cylinder at t=t0
rhocyl = 7900;
eps = 1E-10; % Picard error

% End configuration

% Initialising variables
dx = hx / (mx - 1);
dy = hy / (my - 1);
m = mx * my;
t = 0;

G = E / (2*(1 + nu));
lambda = E*nu / ((1 + nu) * (1 - 2*nu));
Estar = E / ((1 - nu)^2);
mass = pi*R^2 * rhocyl * L; % Mass of cylinder
zcyl = z0cyl;
vcyl = v0cyl;

x = @(i) dx*(mod(i - 1,mx));
y = @(i) dy*floor((i - 1) / mx);
```

```
49  xi = @(i) mod(i − 1,mx) + 1;
50  yi = @(i) floor((i − 1)/mx) + 1;
51
52  % Boundaries:
53  in = (repmat(2:mx−1,my−2,1) + mx∗repmat((1:my−2)',1,mx−2))';
54  b1 = [1:mx,1+mx∗(1:my−1),mx+mx∗(1:my−1)];
55  b2 = 2 + (my−1)∗mx:mx∗my−1;
56  in = in(:);
57  lin = length(in);
58
59  u = zeros(2∗m,1);
60  v = zeros(2∗lin,1);
61  f = sparse(8∗m + 2∗lin,1);
62
63  % Initial conditions:
64  u(:,1) = 0;
65  v(:,1) = 0;
66
67  % Creating the matrix B for eps = B∗u
68  B = sparse(3∗m,2∗m);
69
70  for i = 1:m
71      % Discretisation of eps = 1/2(du/dx + (du/dx)^T)
72      % First eps_11 = du1/dx1
73      if abs(x(i)) < 1e−10; % == 0, ignoring numerical errors
74          % Left boundary
75          B(i, i + 1) = 1/(dx);
76          B(i, i) = −1/(dx);
77      elseif abs(x(i) − hx) < 1e−10;
78          % Right boundary
79          B(i, i) = 1/(dx);
80          B(i, i − 1) = −1/(dx);
81      else
82          % Center
83          B(i, i + 1) = 1/(2∗dx);
84          B(i, i − 1) = −1/(2∗dx);
85      end
86
87      % Now eps_12 = 1/2 (du1/dx2 + du2/dx1)
88      % du1/dx2:
89      if abs(y(i)) <1e−10;
90          % Bottom boundary
91          B(i + m, i + mx) = 1/(dy) / 2;
92          B(i + m, i) = −1/(dy) / 2;
93      elseif abs(y(i) − hy) < 1e−10;
94          % Top boundary
95          B(i + m, i) = 1/(dy) / 2;
96          B(i + m, i − mx) = −1/(dy) / 2;
97      else
98          % Center
99          B(i + m, i + mx) = 1/(2∗dy) / 2;
100         B(i + m, i − mx) = −1/(2∗dy) / 2;
101     end
102     %du2/dx1:
103     if abs(x(i)) < 1e−10;
```

```
104         % Left boundary
105         B(i + m, i + m + 1) = 1/(dx) / 2;
106         B(i + m, i + m) = −1/(dx) / 2;
107     elseif abs(x(i) − hx) < 1e−10;
108         % Right boundary
109         B(i + m, i + m) = 1/(dx) / 2;
110         B(i + m, i + m − 1) = −1/(dx) / 2;
111     else
112         % Center
113         B(i + m, i + m + 1) = 1/(2∗dx) / 2;
114         B(i + m, i + m − 1) = −1/(2∗dx) / 2;
115     end
116
117     % Thirdly, eps_22 = du2/dx2
118     if abs(y(i)) < 1e−10;
119         % Bottom boundary
120         B(i + 2∗m, i + m + mx) = 1/(dy);
121         B(i + 2∗m, i + m) = −1/(dy);
122     elseif abs(y(i) − hy) < 1e−10;
123         % Top boundary
124         B(i + 2∗m, i + m) = 1/(dy);
125         B(i + 2∗m, i + m − mx) = −1/(dy);
126     else
127         % Center
128         B(i + 2∗m, i + m + mx) = 1/(2∗dy);
129         B(i + 2∗m, i + m − mx) = −1/(2∗dy);
130     end
131 end
132
133 % Creating the matrix C for sigma = C eps
134 C = [(lambda + 2∗G)∗speye(m), sparse(m,m), lambda∗speye(m); ...
135       sparse(m,m), 2∗G∗speye(m), sparse(m,m); ...
136       lambda∗speye(m), sparse(m,m), (lambda + 2∗G)∗speye(m)];
137
138 % Given the vector [u;eps;sigma;v], compute d^2u/dt^2 u = T∗sigma for the internal points
139 T = sparse(2∗lin,3∗m);
140 for i = 1:lin
141     j = in(i);
142     T(i,j + 1) = 1/(2∗dx) / rho; % dsigma11/dx1
143     T(i,j − 1) = −1/(2∗dx) / rho;
144     T(i,j + m + mx) = 1/(2∗dy) / rho; % dsigma12/dx2
145     T(i,j + m − mx) = −1/(2∗dy) / rho;
146
147     T(i + lin,j + m + 1) = 1/(2∗dx) / rho; % dsigma12/dx1
148     T(i + lin,j + m − 1) = −1/(2∗dx) / rho;
149     T(i + lin,j + 2∗m + mx) = 1/(2∗dy) / rho; % dsigma22/dx2
150     T(i + lin,j + 2∗m − mx) = −1/(2∗dy) / rho;
151 end
152
153 % Creating the full matrix
154 Asmall = [sparse(2∗m, 8∗m + 2∗lin); ...
155       −B, speye(3∗m), sparse(3∗m, 3∗m + 2∗lin); ...
156       sparse(3∗m, 2∗m), −C, speye(3∗m), sparse(3∗m, 2∗lin); ...
157       sparse(2∗lin, 5∗m), −dtsmall∗T, speye(2∗lin)];
158 Asmall([in;in+m],[in;in+m]) = speye(2∗lin);
```

```
159 Asmall([in;in+m],8*m+1:8*m+2*lin) = -dtsmall*speye(2*lin);
160
161 Abig = [sparse(2*m, 8*m + 2*lin); ...
162       -B, speye(3*m), sparse(3*m, 3*m + 2*lin); ...
163       sparse(3*m, 2*m), -C, speye(3*m), sparse(3*m, 2*lin); ...
164       sparse(2*lin, 5*m), -dtbig*T, speye(2*lin)];
165 Abig([in;in+m],[in;in+m]) = speye(2*lin);
166 Abig([in;in+m],8*m+1:8*m+2*lin) = -dtbig*speye(2*lin);
167
168 % Creating the boundary conditions and right hand vector.
169 for i = b1 % left, bottom, and right boundary
170     % Setting u_x,u_y = 0
171     Asmall(i, i) = 1; % u_x
172     Abig(i, i) = 1;
173     f(i) = 0;
174
175     Asmall(i + m, i + m) = 1; % u_y
176     Abig(i + m, i + m) = 1;
177     f(i + m) = 0;
178 end
179
180 for i = b2 % top boundary
181     % Setting sigma12 = 0, sigma22 = -1
182     Abig(i, i + 6*m) = 1; % sigma12
183     f(i) = 0;
184
185     Abig(i + m, i + 7*m) = 1; % sigma22
186     f(i + m) = -1;
187 end
188
189 % Computing the stress/strains at t = 0:
190 sigma = zeros(3*m,1);
191 data = zeros(8*m+2*lin+2,1);
192 data(end - 1) = z0cyl;
193
194 % Now iterating over time, using Euler Backward
195 j = 2;
196 upwards = 0;
197 while t(j - 1) < t1 && upwards <= 1
198     % Checking which time step we should use
199     if zcyl(j - 1) <= 0 || zcyl(j - 1) + 1.2 * dtbig*vcyl(j - 1) <= 0
200         dt = dtsmall;
201         A = Asmall;
202     else
203         dt = dtbig;
204         A = Abig;
205     end
206
207     % Dynamic allocation of displacement/velocity components
208     u = [u, zeros(2*m,1)];
209     v = [v, zeros(2*lin,1)];
210     t = [t, t(j - 1) + dt];
211     zcyl = [zcyl, 0];
212     vcyl = [vcyl, 0];
213
```

```
214     % Integrating with Euler backwards
215     % Solving using Picard iteration
216     f([in;in+m]) = u([in;in+m],j - 1); % u_{j+1} = u_j + ...
217     f(8*m+1:8*m+2*lin) = v(:,j - 1); % v_{j+1} = v_j + ...
218
219     iter = 1;
220     uprev = u(:,j - 1);
221     vprev = v(:,j - 1);
222     zcylprev = zcyl(j - 1);
223     vcylprev = vcyl(j - 1);
224
225     % Checking where height cylinder < height surface
226     xb = (xi(b2)' - 1) * dx + uprev(b2); % x position of points at surface
227     yb = sqrt(max(0,R^2 - (xb - hx/2).^2)); % height of cylinder wrt x
228     yb(yb == 0) = -Inf;
229     yb = zcylprev - yb + R; % actual height cylinder
230     occurspen = yb <= uprev(b2 + m) & uprev(b2 + m) <= 0;
231
232     while upwards <= 1
233         % Integration z,v cylinder
234         F = pi/4 * Estar * L * max(0, -zcylprev);
235         a = [1, -dt; 0, 1] \ [zcyl(j - 1); vcyl(j - 1) + dt * (F/mass - g)];
236         zcyl(j) = a(1);
237         vcyl(j) = a(2);
238
239         % Applying boundary conditions for top boundary
240         xb = (xi(b2)' - 1) * dx + uprev(b2); % x position of points at surface
241         yb = sqrt(max(0,R^2 - (xb - hx/2).^2)); % height of cylinder wrt x
242         yb(yb == 0) = -Inf;
243         yb = zcyl(j) - yb + R;
244
245         for i = 1:length(b2)
246             % Setting pressure distribution OR displacement
247             A(b2(i), :) = 0;
248             A(b2(i) + m, :) = 0;
249             if occurspen(i)
250                 % Penetration, setting u and v
251                 A(b2(i), b2(i) + 0*m) = 1; % u
252                 f(b2(i)) = 0;
253
254                 A(b2(i) + m, b2(i) + m) = 1; % v
255                 f(b2(i) + m) = yb(i);
256
257             else
258                 % No penetration, setting p_n = 0
259                 % i.e. sigma12 = sigma22 = 0;
260                 A(b2(i), b2(i) + 6*m) = 1; % sigma12
261                 f(b2(i)) = 0;
262
263                 A(b2(i) + m, b2(i) + 7*m) = 1; % sigma22
264                 f(b2(i) + m) = 0;
265             end
266         end
267
268         % Solving the linear system
```

```
269         data = A \ f;
270         u(:,j) = data(1:2*m);
271         epsilon = data(2*m+1:5*m);
272         sigma = data(5*m+1:8*m);
273         v(:,j) = data(8*m+1:8*m+2*lin);
274
275         nrm = norm([u(:,j);v(:,j);zcyl(j);vcyl(j)] - [uprev;vprev;zcylprev;vcylprev]);
276         if nrm < eps
277             % Convergence reached.
278             disp(sprintf('Time step %d (t = %d s ) took %d iterations. Cylinder height: %d m.', j, t(j),
                    iter, zcyl(j)));
279             if ~upwards && vcyl(j) > 0
280                 upwards = 1;
281             elseif upwards && vcyl(j) <= 0
282                 upwards = 2;
283             end
284             break;
285         end
286         if iter > 100
287             disp('Not converging!');
288         end;
289         uprev = u(:,j);
290         vprev = v(:,j);
291         zcylprev = zcyl(j);
292         vcylprev = vcyl(j);
293         iter = iter + 1;
294     end
295     if upwards <= 1
296           j = j + 1;
297     end
298 end
299 end
300
301 % Plotting
302 if 1
303 f = figure;
304 xlabel('x (m)');
305 ylabel('y (m)');
306 X = x((1:m)') - hx/2;
307 Y = y((1:m)') - hy;
308 j = 0:pi/100:2*pi;
309 spherex = R * cos(j);
310 spherey = R * sin(j) + R;
311
312 j = 1;
313 while ~isempty(j)
314     plot(X + u(1:m,j), Y + u(m+1:2*m,j), 'b*');
315     hold on;
316     plot(spherex, spherey + zcyl(j));
317     axis([-hx/2, hx/2, -hy, z0cyl + 2*R]);
318     hold off;
319     pause
320     j = find(t >= t(j) + dtplot, 1);
321 end
322 end
```

A.4 StationaryBeam.m

The following code computes the stationary solution of the beam equation given a pressure distribution p. Both the finite difference as well as the modal approach are applied and compared.

```
% The following solves the differential equation
% 0 = -E*I * (d^4 u / dx^4) + p

% We use a simply supported beam with boundary conditions
% u(0) = u''(0) = u(L) = u''(L) = 0

close all;

% Overal settings:
[L, E, I] = deal(1);

% Method 1: Default finite difference discretisation
% d^4/dx^4 u_i is discretised by (u_(i-2) - 4u_(i-1) + 6u_i - 4u_(i+1) + u_(i+2)) / dx^4
% Discretised equation becomes 0 = A*u + p/mu

% Discretisation settings:
n = 10000;
dx = L / (n - 1);
x = (0:dx:L)';

% Choices for the pressure distribution:
%p = zeros(n, 1);
%p(floor(2/10 * n)) = 3;
%p(floor(9/10 * n)) = -5;
%p = -x.^2 .* (1 - x);
%p = -ones(n,1);
p = -sin(pi*x / L);

% Creating the matrix A
A = sparse(n,n);
%A(2,1:4) = -1 * E * I * [-4, 7, -4, 1]; % Clamped
A(2,1:4) = -1 * E * I * [-2, 5, -4, 1]; % Fixed
for i = 3:n - 2
    A(i, i - 2 : i + 2) = -1 * E * I * [1, -4, 6, -4, 1];
end
%A(n - 1, n - 3:n) = -1 * E * I * [1, -4, 7, -4]; % Clamped
A(n - 1, n - 3:n) = -1 * E * I * [1, -4, 5, -2]; % Fixed
A = A / dx^4;

A(1,1) = 1;
p(1) = 0; % Setting u = 0 on boundary
A(n,n) = 1;
p(n) = 0;

% Solving the linear system
u = -A \ p;


```

```
49 % Method 2. Using modes
50 % Modes are given by : u_n(x) = 2 / L * sin(n pi x / L)
51
52 % Settings:
53 k = 4; % Amount of nodes
54
55 % Computing the Riemann integral
56 int = zeros(k, 1);
57 for i = 1 : k
58     int(i) = dx * sum(sqrt(2/L) * sin(i * pi * x / L) .* p);
59 end
60
61 % Computing the modal coefficients c
62 c = 1 / (E*I) * (L ./ ((1:k)' * pi)).^4 .* int;
63
64 % Computing the modal solution
65 um = zeros(n, 1);
66 for i = 1 : k
67     um = um + c(i) * sqrt(2/L) * sin(i * pi * x / L);
68 end
69
70
71 % Plotting
72 plot(x, u, 'b-', x, um, 'r-.');
73 legend('Finite difference', 'Modes');
74 hold off;
75
76 % Computing the error
77 norm(um - u)
```

A.5 FullBridgeModes.m

This is the Matlab code simulating train-bridge contact. Global deformation is computed using modes and combined with the quasi-static results of CONTACT. The Quasi-Newton approach is applied and combined with the Backward Euler scheme.

```
% The following program simulates the deformation that occurs
% for a train moving over a bridge.
% This approach uses modes and is combined with the quasi−static
% results computed using CONTACT.

if 1
% Properties of the bridge
L = 20;
B = 8;
E = 2E+11;
nu = 0.50; % Poisson ratio
I = 0.0104;
mu = 7900;

% Properties of the cylinder
rho = 7900;
R = 0.3;
veloc = 30;

% Other settings
g = 9.81; % Gravitational acceleration
m = 30; % Amount of modes
nxplot = 500; % Only used for plotting
nxc = 25;
ny = 11;
dt = 0.001;
plotdt = 0.01;
t1 = 0.67;
eps = 1e−5;

% Dynamic variables
omega = 0; % Relaxation, 0 = none
xif = 0; % Relaxation of derivative. 0 = none
micro = 1e−7; % For finite difference derivative

dxc = 2∗R / (nxc − 1);
dxplot = L / (nxplot − 1);
dy = B / (ny − 1);
y = (−B/2:dy:B/2)';
t = (0:dt:t1)';
k = length(t);
c = zeros(2∗(m + 1), k);
deltav = zeros(k, 1);
Fv = zeros(k, 1);
pos = 0 + veloc ∗ t;
xcyl = (−R:dxc:R)';
zcyl = R − sqrt(max(0, R^2 − xcyl.^2));
xplot = (0:dxplot:L)';
```

```
49  mc = rho * pi*R^2 * B; % Mass of cylinder
50
51  % For contact
52  settings.G = E / (2*(1 + nu));
53  settings.nu = nu;
54  settings.dx = dxc;
55  settings.dy = dy;
56
57  % Modes
58  w = @(j,x) sqrt(2/L) * sin(j * pi * x / L);
59  beta = @(j) j*pi/L;
60  lambda = @(j) beta(j).^2 * sqrt(E*I/rho);
61
62
63  if 1
64  % Euler Backwards
65  % Creating the matrix D
66  D = zeros(2*m + 2);
67  for i = 1:m
68      D(2*i - 1 : 2*i, 2*i - 1 : 2*i) = [1, -dt;
69                  dt * E*I/rho * beta(i)^4, 1];
70  end
71  D(2*m + 1:end, 2*m + 1:end) = [1, -dt;
72                                 0, 1];
73  for j = 2:k
74      fclose('all');
75      % Solving the implicit system using Picard iteration for the rhs
76      cprev = c(:, j - 1);
77      iter = 1;
78
79      while 1
80          before = tic;
81          xi = 1 + xif^iter;
82
83          disp(['Time step ', num2str(j),'/', num2str(k),' iteration ', num2str(iter), '.']);
84
85          % Creating the matrix B
86          B = D;
87
88          % First, we run CONTACT
89          % Computing the height of the bridge wrt x points on wheel at time step k + 1
90          x = pos(j) + xcyl;
91          zbridge = zeros(nxc, 1);
92          for i = 1:m
93              zbridge = zbridge + cprev(2*i - 1) * w(i, x);
94          end
95
96          % The right hand side:
97          rhs = c(:, j - 1);
98
99          % Computing the penetration
100         pen = (zcyl + cprev(2*m + 1)) - zbridge;
101
102         a1 = x(max(1, find(pen < 0, 1) - 1));
103         a2 = x(min(nxc, find(pen < 0, 1, 'last') + 1));
```

```
104
105     if isempty(a1) || isempty(a2)
106         % No penetration
107         dx = dxc;
108         settings.dx = dxc;
109
110         [delta, xpos] = min(pen);
111         x0 = x(xpos);
112         pen = pen - delta; % Acting as if penetration is exactly 0
113         cprev(end - 1) = cprev(end - 1) - delta;
114     else
115         % There is penetration, changing mesh size
116         dx = (a2 - a1) / (nxc - 1);
117         settings.dx = dx;
118         x = (a1:settings.dx:a2)';
119
120         % Computing height of the bridge for new x
121         zbridge = zeros(nxc, 1);
122         for i = 1:m
123             zbridge = zbridge + cprev(2*i - 1) * w(i, x);
124         end
125
126         % Computing penetration for new x
127         pen = ((R - sqrt(max(0, R^2 - (x - pos(j)).^2))) + cprev(2*m + 1)) - zbridge;
128                         %zcyl + height cyl - height bridge
129
130         [delta,xpos] = min(pen(:));
131         x0 = x(xpos);
132     end
133     deltav(j) = -delta;
134
135     % Repmat to use for CONTACT
136     pen = repmat(pen', ny, 1);
137
138     % Running CONTACT and computing the pressure distribution and contact force
139     [Fc,p,l] = run_contact(pen, settings);
140     p = -p(round(size(p,1) / 2), :)';
141     Fv(j) = Fc;
142
143     [Fc2,p2] = run_contact(pen - micro, settings);
144     p2 = -p2(round(size(p2,1) / 2), :)';
145
146     if Fc == Fc2
147         disp('WARNING: d F / d delta = 0. Making micro larger.');
148         micro = micro * 10;
149         pause(3);
150         continue;
151     end
152
153     % Settings the fpenratio
154     fpenratio = (Fc2 - Fc) / micro;
155
156     % For each mode, we compute the integrals p(x,t) w(x) d x and p2(x,t) w(x) dx
157     % The derivative w.r.t. delta is computed
158     % The matrix B
```

```
159         for i = 1:m
160             int = dx * sum(p .* w(i, x));
161             int2 = dx * sum(p2 .* w(i, x));
162             intpenratio = (int2 - int) / micro;
163 
164             B(2*i, 2*m + 1) = dt/rho * xi * intpenratio;
165             for i2 = 1:m
166                 B(2*i, 2*i2 - 1) = B(2*i, 2*i2 - 1) - dt/rho * xi * intpenratio * w(i2, x0);
167             end
168             rhs(2*i, 1) = rhs(2*i, 1) + dt * 1/rho * (int + xi * intpenratio * (cprev(2*m + 1) - zbridge(
                    xpos)));
169         end
170 
171         % For the height of the cylinder:
172         for i = 1:m
173             B(end, 2*i - 1) = -dt * xi * fpenratio * w(i, x0) / mc;
174         end
175 
176         % For the height of the cylinder continued:
177         B(end, end - 1) = dt * xi * fpenratio/mc;
178         rhs(end, 1) = rhs(end, 1) + dt * ((Fc + xi * fpenratio * (cprev(2*m + 1) - zbridge(xpos)))/
                mc - g);
179 
180         % Finally, applying Picard iteration by solving the system
181         cnewact = B \ rhs;
182 
183         % Using under relaxation
184         cnew = (1 - omega) * cnewact + omega * cprev;
185 
186         disp(['The iteration took ',num2str(toc(before)),' seconds.']);
187         % Testing for convergence
188         if norm(cnewact - cprev) < eps
189             disp(['Time step ', num2str(j), ' converged after ', num2str(iter), ' iterations.']);
190             c(:, j) = cnew;
191             break;
192         end
193 
194         if iter >= 100
195             disp('Picard iteration does not converge.');
196             return;
197         end
198 
199         cprev = cnew;
200         iter = iter + 1;
201 
202     end
203 end
204 end
205 
206 end
207 
208 
209 % Computing the solution
210 u = zeros(nxplot, k);
211 for j = 1:k
```

```
212     for i = 1:m
213         u(:, j) = u(:, j) + c(2*i - 1, j) * w(i, xplot);
214     end
215 end
216
217
218 % Plotting
219 if 1
220 f = figure;
221 xlabel('x (m)');
222 ylabel('z (m)');
223 set(gca, 'ydir','reverse');
224 for i = 1:max(1,round(plotdt/dt)):k
225     plot(xplot,u(:,i), pos(i) + xcyl, zcyl + c(2*m + 1, i));
226     axis([0,L,min(u(:)),max(u(:))]);
227     pause;
228 end
229 end
```