MASTER'S THESIS

Technische Universität München

Chair of Structural Analysis

Computational Mechanics

Formulation and object-oriented implementation of a nonlinear node-to-surface mechanical contact algorithm

Author:

Dipl.-Ing. (FH) Stefan SICKLINGER

Advisor:

Univ. Prof. Dr.-Ing. Kai-Uwe BLETZINGER

Supervisors:

Dr.-Ing. Roland WÜCHNER

Dipl.-Ing. (FH) Matthias FIRL, M. Sc.

Munich, 12. September 2010

18. Dezember 2012

Ich versichere, dass ich diese Masterarbeit selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel verwendet habe.

München, 12. September 2010 Dipl.-Ing. (FH) Stefan Sicklinger

Acknowledgments

It is a pleasure to thank those people who made this thesis possible.

At first I would like to thank Prof. Dr.-Ing. K.-U. Bletzinger for offering me the great opportunity to do my master's thesis at the chair of Structural Analysis in a subject especially catered to me.

I owe my deepest gratitude to my supervisor Dr.-Ing. R. Wünchner for his encouragement, for his excellent support and not to forget for the many gainful professional discussions during my project.

I gratefully acknowledge my second supervisor Dipl.-Ing. (FH) M. Firl, M.Sc. for his excellent support and his helpful advices especially during the implementation process.

Special thanks goes to my coworkers at SIMULIA Germany for many interesting professional discussions and helpful hints for my thesis.

I express my profound appreciation to my girlfriend Daniela for her encouragement, understanding and patience even during the hard times of this thesis.

Last, I want to thank my parents, without whom I would never have been able to achieve so much.

Abstract

Contact problems play a dominate role in almost all engineering disciplines, thus there is a huge demand for computational contact mechanics. One of the most impressive application is crash analysis. In the scope of this master's thesis contact mechanics is approached in an object-oriented manner. A three-dimensional nonlinear node-to-surface contact algorithm is derived and implementation details are presented. For constraint enforcement the penalty method is used.

The thesis is composed of six chapters. The first two chapters give an introduction to the topic and describe computational contact mechanics in the framework of object orientation. Next, the basic theory is presented. The implemented 4-node contact element is formulated in the fourth chapter. After the contact element is derived the object-oriented implementation is explained. Finally, examples are presented and discussed.

Keywords: computational contact mechanics, object-oriented, node-to-surface, penalty method, surface-coupled, three-dimensional, frictionless

Contents

Chapter 1

Introduction and motivation

This master's thesis deals with computational contact mechanics as a subclass of surface-coupled problems. Contact problems arise almost everywhere in nature and they are crucial in engineering. No car could drive without frictional contact and even walking would be impossible without friction. Mechanical contact problems are also key problems in safety analysis e.g. in car crashworthiness, where the crash analysis is a highly complicated contact problem (see figure 1.1). Contact is also dominate in medicine e.g. placing a stent inside an artery. Consumer goods packaging maybe is not an obvious application of contact mechanics, but the packaging must survive drop tests. Simulations are performed in advance to optimize the packaging, so the key feature of these simulations is a robust contact algorithm. One problem is therefore to design and derive efficient, accurate and robust contact algorithms, which can be used in the framework of continuum me-

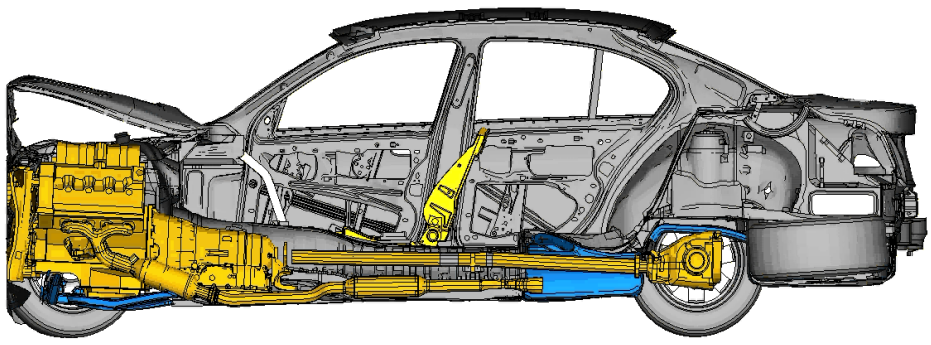


Figure 1.1: BMW 5 series front crash half model

chanical simulations. In the scope of this thesis a three-dimensional, nonlinear contact algorithm is derived. For constraint enforcement the penalty method is used, the contact interface was discretized by a node-to-surface formulation. The contact problems are solved within the framework of a nonlinear static analysis.

1.1 Aim of the master's thesis

The main aim of the presented work is to approach computational contact mechanics in an object-oriented manner. Contact mechanics can, as a subclass of surface-coupled problems, be decomposed into components. The identification of these components is the key to object-oriented programming.

Within this master's thesis a three-dimensional node-to-surface frictionless contact algorithm is formulated and implemented in order to demonstrate the object-oriented concept.

1.2 Overview

In this chapter only a short introduction to the topic and the motivation for computational contact mechanics are given. The following chapter addresses the issue of how a computational contact algorithm can be decomposed in an object-oriented manner and identifies components of computational contact algorithms. The third chapter presents the basic theory. An introduction to linear and nonlinear FEM is provided. Furthermore some simple one-dimensional mechanical contact examples are presented. As the examples in chapter 3 are one-dimensional, chapter 4 expands the presented concepts to three dimensions and a 4-node contact element is derived. For the constraint enforcement the penalty method is used exclusively. The whole object-oriented contact algorithm is discussed in detail in chapter 5. Finally, examples are presented and discussed in chapter 6.

Chapter 2

Surface-coupled problems and object-orientation

The whole contact algorithm is designed in an object-oriented manner, so an object-oriented programming language is needed for the implementation. As a "middle-level" language C++ provides a good compromise of efficiency and high-level features like object-oriented programming and data abstraction. Furthermore C++ is widely used and has good support for parallel programming. It is therefore a good choice for FE software.

The design philosophy for computational contact mechanics is such that it can easily be expanded to different contact formulations and enforcement methods. As mechanical contact is a subclass of surface-coupled problems, parts of the implementation can be reused for different other surface-coupled problems like fluid-structure interaction (FSI) or thermal fluid-structure interaction (TFSI).

Contact mechanics is a nonlinearity in the boundary conditions, thus it is a surface-coupled problem. The computational power of an average PC is around 5 GFlops at the moment. The tremendous increase in computational power during the last years makes it possible to simulate multiphysical problems. Most of these problems are coupled by means of surfaces. Because these problems share the feature of quantity exchange over the surface, it is attempted to implement the contact algorithm as general as possible in order to be able to expand it to other surface-coupled problems later.

Siehe [3]

2.1 Fluid-structure interaction

Where fluid flow causes deformation of a structure, fluid-structure interaction occurs. The resulting deformation is linked to the pressure load from the fluid flow. The deformation of the structure in turn has an influence on the fluid flow. According to [40, chapter 18], fluid-structure interaction is a class I problem. This class contains coupled problems in which coupling occurs on domain interfaces via the boundary conditions imposed there. In general, different discretizations can be used for the different domains. The domains may be physically different

or similar. In the case of fluid-structure interaction it is possible to use finite volume methods for the fluid domain and finite elements for the structural domain. The pressure and displacement fields are interchanged at the interface . In this point fluid-structure interaction is similar to a frictionless contact problem. This similarities can be exploited by object-oriented modeling.

2.2 Thermal fluid-structure interaction

Thermal fluid-structure interaction handles the heat transport in both domains (structural and fluid) in addition to fluid-structure interaction. Therefore thermal fluid-structure interaction is a class I and class II problem at the same time: Class II coupled problems are problems in which the various domains overlap. The coupling in class II occurs through the governing differential equations [40]. Both the structure and the fluid domain are coupled to the heat transfer by the governing differential equations. In contrast to fluid-structure interaction there are three fields to interchange at the interface.

2.3 Contact problems

The field of contact mechanics is a wide field. It is already quite challenging to define the problem. Figure 2.1 on the next page shows an overview of a general contact algorithm and gives insight into the possibilities of the contact problem definition. We start with the determination of the contact pairing as a member of the primary layer of the contact problem. The contact searching, which is done to determine the contact pairs, can be divided into three main categories. The simplest approach requires the user to specify the assumed contact partners. This approach is used in this master's thesis. The usual way is to specify a master and a slave surface. From this information, a fairly simple contact search algorithm can then determine the pairings. Of course there are even simpler approaches like gap elements, but they are a poor representation of contact, because the pairings can never change. For a car crash simulation even the surface-based contact definition would be a tedious task, hence more sophisticated approaches have been developed. One of them is "general contact", for which the user just needs to switch contact on. The pairings are then determined automatically by the algorithm (see [30]). These techniques can also be used if self contact occurs.

A second crucial point in the problem definition addresses the issue of which contact behaviors should exactly be modeled. In nature contact is never a "hard" phenomena. Because of the roughness of the contact surfaces contact happens in a smooth way: Unless the micro-structure of the contact surfaces is flat, contact is only partially achieved (see also chapter 5). In order to be able to take this into account, one way is to give the user the possibility to specify a force-displacement curve. Another application for this "soft" contact is the simulation of bodies coated with paint. Measurement data could be used in order to provide realistic force-displacement curves, so there is no need to model the paint layer with elements. These are some applications for the physical contact behavior in

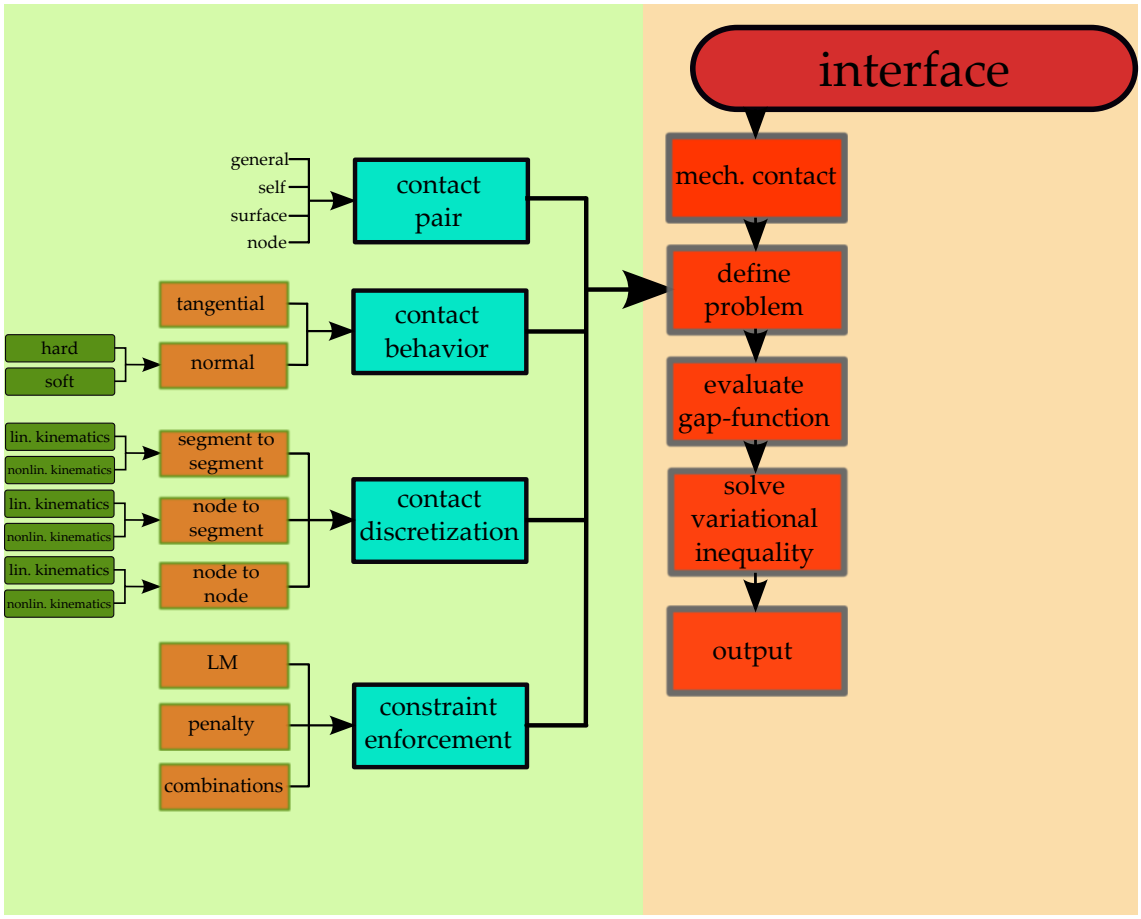


Figure 2.1: Components of computational contact

normal direction. For the tangential behavior there are even more: Different friction models can be used, some of them can even account for wear.

In the third and fourth component (see cyan blocks in figure 2.1) mathematics plays the dominate role. The simplest approach to discretize the contact is the node-to-node approach. The drawback of the simple contact discretization is that it works only with conforming meshes. The method used in this thesis is the node-to-surface discretization, which can handle non-matching grids. More advanced techniques are surface-to-surface discretizations (e.g. mortar method). With each of the discretizations a combination with linear and nonlinear kinematics is possible.

The discretization methods are strongly linked to the constraint enforcement methods. These deal with the question how to model the constraint in the functional, which represents the structural problem. Most methods stem from the optimization theory. Some of the most known are penalty, Lagrange multipliers and augmented Lagrange techniques. Only the penalty method will be discussed in this work. Penalty methods cannot enforce the constraint exactly, they always allow slight penetrations of the contact partners. In theory it would be possible to enforce the constraint exactly, even with penalty methods, but therefore an infinitely large penalty stiffness would be needed. If the penalty stiffness is too large, the overall equation system becomes ill-conditioned. An estimate for an

optimal penalty stiffness is given in [20]. The advantages of penalty methods are that they are not too complicated to implement and, in combination with good solution strategies, they show quite good convergence.

In the following three chapters the implemented contact algorithm will be derived from a theoretical point of view. We will start with the basics of linear finite elements and then develop step by step the three-dimensional frictionless node-to-surface contact algorithm with nonlinear kinematics. After the theory has been presented the object-oriented implementation will be discussed.

2.4 Object-orientation for computational contact mechanics

With the help of the previous discussion it is now possible to decompose computational contact mechanics in an object-oriented manner. The first component is the problem definition (see figure 2.1 on the previous page), which has four subcomponents. These four subcomponents (contact pair, contact behavior, contact discretization and contact enforcement) build the basis for the class structure of an object-oriented implementation of computational contact mechanics.

These four objects are accessible by the contact element class (see also figure 5.1 on page 34), which evaluates the gap function. These five objects can be transformed into a class structure. It is not necessary to change the structure of the underlying object-oriented finite element software (in this case Carat++). If it is not possible to use the standard nonlinear solution scheme (e.g. Newton-Raphson) in combination with the contact formulation, the structure of the finite element software needs to be slightly adapted. In that case the solution methods should be separated, so it is possible to interchange them according to the need of the analysis type and the contact formulation.

Chapter 3

Concise theory

In this chapter some methods for solving computational contact problems are presented. For minimizing the unconstrained functional, the finite element method is used. First it is introduced in a linear setting, later it is generalized to nonlinear problems. Minimizing the unconstrained functional is equal to solving the operational form of continuum mechanics problems. After the finite element method has been derived, an introduction on the nature of mechanical contact problems is given. Furthermore, algorithms are discussed that can be used in combination with the finite element method to solve mechanical contact problems.

3.1 The finite element method for solving PDEs

A famous numerical method for solving partial differential equations (PDEs) is the so called finite element method. The term "finite element" was first used by Clough [26]. In the following some insight in the method will be given by using one of the simplest examples in mechanics, the linear elastic truss. Finite elements are based on the weak form of a problem, which is equal to the strong form (proof see [40]). Let us examine the finite element method with the help of an example.

3.1.1 Linear hanging truss

The truss has a constant cross sectional area $A_0 = 1\,\mathrm{m}^2$ and a length of $L = 1\,\mathrm{m}$. The material is linear elastic with uniform density $\rho = 1\,\mathrm{kg/m^3}$. The gravity body load can be converted in a pseudo line load, which is given by $n = A_0 \rho a$. By the help of force balance and Taylor series one can derive the governing ordinary differential equation for the normal force $N(x)$ in the truss (see figure 3.1 on the next page).

$$\frac{\mathrm{d}N(x)}{\mathrm{d}x} = -n \tag{3.1}$$

Note that this equation is derived under the geometric linear assumption.

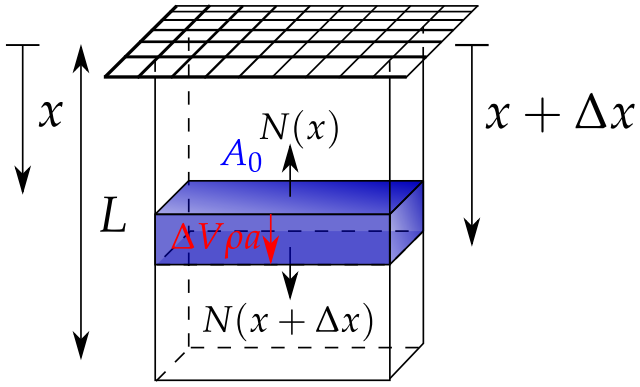


Figure 3.1: Hanging truss subjected to gravity

3.1.1.1 Strong form

With the material law for linear elastic material ($\sigma = E\epsilon = {}^{N(x)}/_{A_0}$) and the kinematic relation ($\epsilon^{eng} = {}^{\mathrm{d}u(x)}/_{\mathrm{d}x}$) one can state the second order differential equation for the hanging truss by

$$EA_0 \frac{\mathrm{d}^2 u(x)}{\mathrm{d}x^2} = -n \tag{3.2}$$

where $u(x)$ is the displacement of the truss. The displacements are assumed to be small. Note that equation (3.2) is called strong form or operator form. The problem has the following boundary conditions:

$$u(0) = 0 \tag{3.3}$$

$$\sigma(1) = 0 \rightarrow \left.\frac{\mathrm{d}u(x)}{\mathrm{d}x}\right|_{x=1} = 0 \tag{3.4}$$

3.1.1.2 Weak form

Equation (3.2) can be written in the form

$$Ou + n = 0 \tag{3.5}$$

where $O := EA_0 {}^{\mathrm{d}^2(x)}/_{\mathrm{d}x^2}$ is called operator. The weak form can be written with the inner product $(u, w) := \int_0^1 u(x)w(x)\mathrm{d}x$ as

$$(Ou, w) + (n, w) = 0 \tag{3.6}$$

$$\int_0^1 EA_0 \frac{\mathrm{d}^2 u(x)}{\mathrm{d}x^2} w(x)\mathrm{d}x = -\int_0^1 nw(x)\mathrm{d}x \tag{3.7}$$

with $w(x)$ being the so-called weighting function. The weighting functions in mechanics correspond to the virtual displacements δu. By applying integration

by parts (note that in higher dimensions Green's formula is needed) we gain the weak form (equation (3.9)).

$$\int_0^1 EA_0 \frac{\mathrm{d}^2 u(x)}{\mathrm{d}x^2} w(x)\mathrm{d}x = -EA_0 \int_0^1 \frac{\mathrm{d}u(x)}{\mathrm{d}x}\frac{\mathrm{d}w(x)}{\mathrm{d}x}\mathrm{d}x + EA_0 \left[\frac{\mathrm{d}u(x)}{\mathrm{d}x} w(x)\right]_0^1 \quad (3.8)$$

$$EA_0 \int_0^1 \frac{\mathrm{d}u(x)}{\mathrm{d}x}\frac{\mathrm{d}w(x)}{\mathrm{d}x}\mathrm{d}x = \int_0^1 nw(x)\mathrm{d}x \quad (3.9)$$

The term in equation (3.8) is zero because one requires $w(0) = 0$ and $u'(1) = 0$. The latter is called natural boundary condition because it is fulfilled automatically.

Heading towards a numerical solution we need to replace the infinite function spaces for $u(x) \in S$ and $w(x) \in W$ by a finite-dimensional approximation, which are subspaces of the infinite function spaces.

3.1.1.3 Bubnov-Galerkin method

Now we discretize the weak form and arrive at the Galerkin form of the problem. The Galerkin form is a special form of weighted residual methods. Furthermore we restrict ourselves to the Bubnov-Galerkin method, which produces symmetric stiffness matrices in combination with second-order differential equations. The Galerkin method discretizes the weak form. The Galerkin form can be stated as

$$EA_0 \int_0^1 \frac{\mathrm{d}u^h(x)}{\mathrm{d}x}\frac{\mathrm{d}w^h(x)}{\mathrm{d}x}\mathrm{d}x = \int_0^1 nw^h(x)\mathrm{d}x \quad (3.10)$$

where $u^h(x) \in S^h \subset S$ and $w^h(x) \in W^h \subset W$ are now elements of the finite dimensioned function subspaces of S and W, respectively. In Bubnov-Galerkin methods the trial space S^h and the test space W^h are spanned by the same basis $\phi_1(x), ..., \phi_N(x)$. $u^h(x)$ is composed by $u^h(x) = \sum_{i=1}^N \phi_i(x) u_i$, where u_i are the nodal values.

For the hanging truss example we choose a linear three element approximation. Linear means, that we choose local hat functions for each of the four nodes (see figure 3.2 on the following page). The approximation for $u(x)$ is a linear function, which is C^0-continuous at the nodes.

The Galerkin form for a representation with three linear element (two nodes each) reads

$$EA_0 \sum_{j=1}^N \int_0^1 \frac{\mathrm{d}\phi_j(x)}{\mathrm{d}x}\frac{\mathrm{d}\phi_i(x)}{\mathrm{d}x}\mathrm{d}x\, u_j = \int_0^1 n\phi_i(x)\mathrm{d}x \quad (3.11)$$

where $i = 1, ..., 4$. The domain integral can be splitted at each element border, hence the Galerkin form is

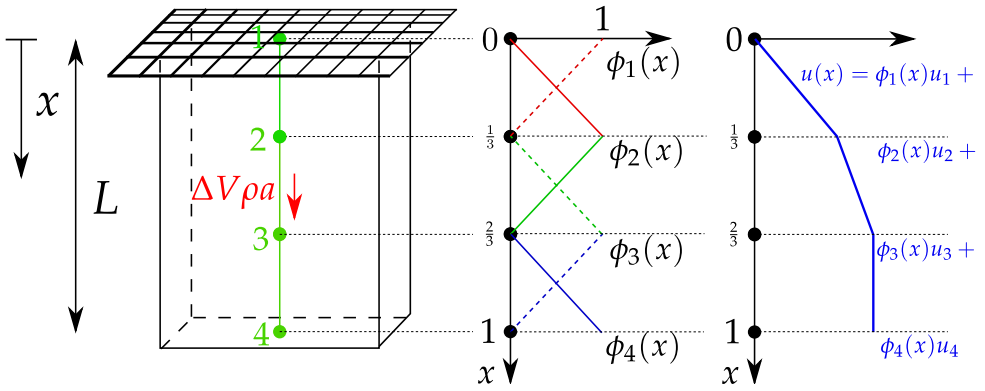


Figure 3.2: Linear approximation for the hanging truss

$$
\begin{aligned}
& EA_0 \sum_{j=1}^{2} \int_0^{1/3} \frac{\mathrm{d}\phi_j(x)}{\mathrm{d}x} \frac{\mathrm{d}\phi_i(x)}{\mathrm{d}x} \mathrm{d}x \, u_j \\
+ & EA_0 \sum_{j=2}^{3} \int_{1/3}^{2/3} \frac{\mathrm{d}\phi_j(x)}{\mathrm{d}x} \frac{\mathrm{d}\phi_k(x)}{\mathrm{d}x} \mathrm{d}x \, u_j \\
+ & EA_0 \sum_{j=3}^{4} \int_{2/3}^{1} \frac{\mathrm{d}\phi_j(x)}{\mathrm{d}x} \frac{\mathrm{d}\phi_l(x)}{\mathrm{d}x} \mathrm{d}x \, u_j \\
= & \int_0^{1/3} n\phi_i(x) \mathrm{d}x + \int_{1/3}^{2/3} n\phi_k(x) \mathrm{d}x + \int_{2/3}^{1} n\phi_l(x) \mathrm{d}x
\end{aligned}
\tag{3.12}
$$

where $i = 1, ..., 2$, $k = 2, ..., 3$ and $l = 3, ..., 4$. Equation (3.12) represents four linear equations with four unknowns (u_j), hence all the individual integrals can be written in a matrix (element matrix) and in a vector (element load vector), respectively. In the finite element method the integration is done numerically (e.g. Gauss integration), but in this example we stick to analytical integration.

The element stiffness matrix is computed as follows by assuming that $E = 1\,\mathrm{N/m^2}$ and $a = 1\,\mathrm{m/s^2}$. (Note that all the three element stiffness matrices and element load vectors are the same, therefore only one candidate is stated (with $\phi_1(x) = 1 - 3x$ and $\phi_2(x) = 3x$)).

$$
\mathbf{K}^{e1} = \mathbf{K}^{e2} = \mathbf{K}^{e3} = EA_0 \int_0^{1/3} \frac{\mathrm{d}\phi_j(x)}{\mathrm{d}x} \frac{\mathrm{d}\phi_i(x)}{\mathrm{d}x} \mathrm{d}x = \begin{bmatrix} 3 & -3 \\ -3 & 3 \end{bmatrix} \tag{3.13}
$$

Note that these stiffness matrices could also be derived by a linearization of equation (3.12) around $\mathbf{u}$. As equation (3.12) is linear in $\mathbf{u}$, the derivative $\partial/\partial\mathbf{u}$ of equation (3.12) does not depend on $\mathbf{u}$ anymore.

$$
\mathbf{f}^{e1} = \mathbf{f}^{e2} = \mathbf{f}^{e3} = \int_0^{1/3} n\phi_i(x) \mathrm{d}x = \begin{bmatrix} \frac{1}{6} \\ \frac{1}{6} \end{bmatrix} \tag{3.14}
$$

The assembled system for three connected elements representing the hanging truss is given by

$$\mathbf{Ku} = \mathbf{f} \tag{3.15}$$

$$\begin{bmatrix} 3 & -3 & 0 & 0 \\ -3 & 6 & -3 & 0 \\ 0 & -3 & 6 & -3 \\ 0 & 0 & -3 & 3 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = \begin{bmatrix} \frac{1}{6} \\ \frac{1}{3} \\ \frac{1}{3} \\ \frac{1}{6} \end{bmatrix} \tag{3.16}$$

The system is not solvable yet, because of the missing boundary condition at the top of the truss $u(0) = 0$, which means that $u_1 = 0$. This renders the sparse linear system to:

$$\begin{bmatrix} 6 & -3 & 0 \\ -3 & 6 & -3 \\ 0 & -3 & 3 \end{bmatrix} \begin{bmatrix} u_2 \\ u_3 \\ u_4 \end{bmatrix} = \begin{bmatrix} \frac{1}{3} \\ \frac{1}{3} \\ \frac{1}{6} \end{bmatrix} \tag{3.17}$$

In figure 3.3 the analytical solution is compared with the linear finite element solution. The FEM solution matches the analytical one exactly at the nodes, but this is unfortunately not the case in general.

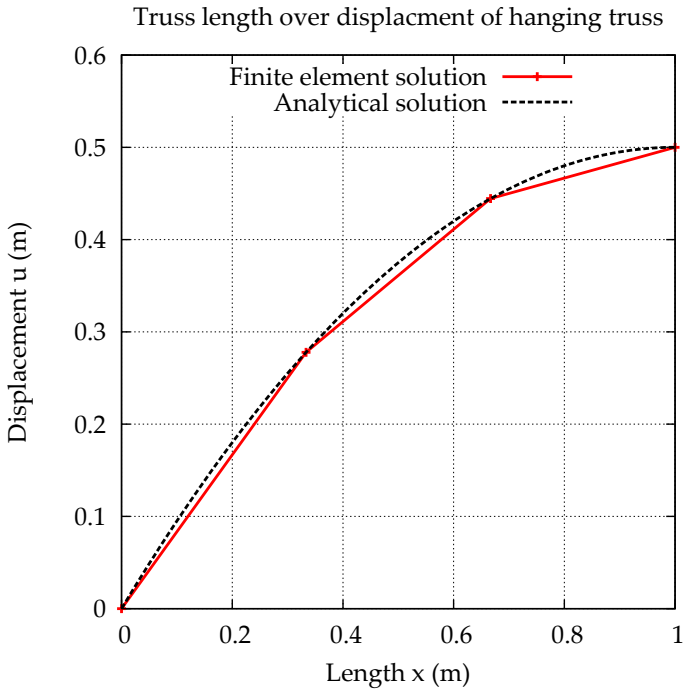


Figure 3.3: Numerical and analytical solution to the hanging linear truss

3.1.1.4 Variational form (functional)

First we should recall the introduced notation in equation (3.5) on page 8. The quadratic functional can now be written by:

$$\begin{aligned}\Pi(w) &= \frac{1}{2}(Ow, w) + (n, w) \\ &= EA_0 \int_0^1 - \left(\frac{\mathrm{d}w(x)}{\mathrm{d}x}\right)^2 \mathrm{d}x + EA_0 \Big[\frac{\mathrm{d}w(x)}{\mathrm{d}x} \overset{0}{\cancel{w(x)}}\Big]_0^1 + \int_0^1 nw(x)\mathrm{d}x\end{aligned} \tag{3.18}$$

This is the functional to be minimized. We assume that $u(x)$ is the function which renders equation (3.18) to a minimum. If $u(x)$ is a stationary point of the functional, the following must hold [8, 9, 25]:

$$\Pi(u) \leq \Pi(u + \iota w) = \Pi(u) + \iota\left[(Ou, w) + (w, n)\right] + \frac{1}{2}\iota^2(Ow, w) \tag{3.19}$$

This can be derived under the assumption that O is linear, $u(x)$ and $w(x)$ are real and ι is also real. Since equation (3.19) holds for ι on both sides of zero , the linear term (first variation in the following denoted by δ) must vanish. This is equivalent to:

$$(Ou, w) + (w, n) = 0 \tag{3.20}$$

$$\int_0^1 EA_0 \frac{\mathrm{d}u(x)}{\mathrm{d}x}\frac{\mathrm{d}w(x)}{\mathrm{d}x}\mathrm{d}x - \int_0^1 nw(x)\mathrm{d}x = 0 \tag{3.21}$$

$$\int_0^1 EA_0 \frac{\mathrm{d}u(x)}{\mathrm{d}x}\frac{\mathrm{d}w(x)}{\mathrm{d}x} - nw(x)\mathrm{d}x = 0 \tag{3.22}$$

By comparison of equation (3.22) and equation (3.9) on page 9 the weak form is identified again (weak extremum).

If integration by parts is again applied on equation (3.22), and the boundary terms are canceled, one arrives at:

$$-\int_0^1 EA_0 \frac{\mathrm{d}^2u(x)}{\mathrm{d}x^2} - nw(x)\mathrm{d}x = 0 \tag{3.23}$$

$$\int_0^1 EA_0 \frac{\mathrm{d}^2u(x)}{\mathrm{d}x^2} + nw(x)\mathrm{d}x = 0 \tag{3.24}$$

This is true if the integrand is zero, which gives the initial strong form of the problem, also know as Euler's equation:

$$EA_0 \frac{\mathrm{d}^2u(x)}{\mathrm{d}x^2} = -n \tag{3.25}$$

This now completes the circle. In the following we will focus on constrained extrema of functionals in contrary to the unconstrained ones from this section. Note that it can be shown that for any quadratic functional a corresponding Euler equation (strong form) can be established, but the converse is unfortunately not true, because only certain forms of differential equations can be identified as Euler equations of quadratic functionals [38].

3.1.2 Geometrically nonlinear hanging truss

In the following section the hanging truss problem is modified a little bit. Just one element is used to represent the hanging truss (see figure 3.4). The linear solution to the modified problem is also $u_2 = 0.5\,\mathrm{m}$. The force vector is now a scalar and reads $f = 0.5\,\mathrm{N}$, but as we see later, we need to reduce it to a corresponding gravity load of $a = 0.5\,\mathrm{m/s^2}$, which leads to the the force vector $f = 0.25\,\mathrm{N}$. As an engineering strain ϵ^{eng} of 0.5 violates the small strain assumption, we need to extend the hanging truss example to finite deformations (finite strain).

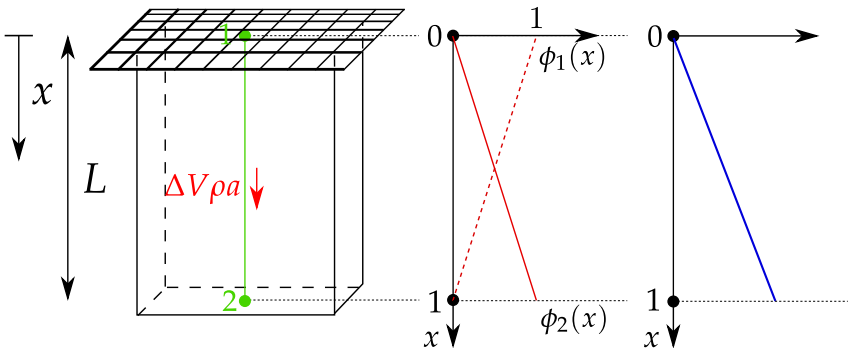


Figure 3.4: Hanging truss subjected to gravity (nonlinear formulation)

3.1.2.1 Nonlinear strain measures

There are many strain measures for nonlinear problems [7, 13, 17, 21, 33]. Here just two of them will be discussed.

$$\epsilon^{GL} = \frac{1}{2}\frac{l^2 - L^2}{L^2} \tag{3.26}$$

$$\epsilon^{ln} = ln(\lambda) = ln\left(\frac{l}{L}\right) = ln\left(1 + \epsilon^{eng}\right) \tag{3.27}$$

The strain measure ϵ^{GL} is called Green-Lagrange strain, ϵ^{ln} is called Hencky strain or logarithmic strain. Both strain measures are well suited for large rotation problems, as both produce zero strain for large rigid body motions (rotations/displacements) in higher dimensions. But only the Hencky strain gives reasonable results for large strain problems, as will be shown by an example. As we still stick

to a linear elastic material law, we do not need to bother about work conjugated strain-stress measures, so we can simply replace the stress by the multiplication of the Young's modulus with the strain.

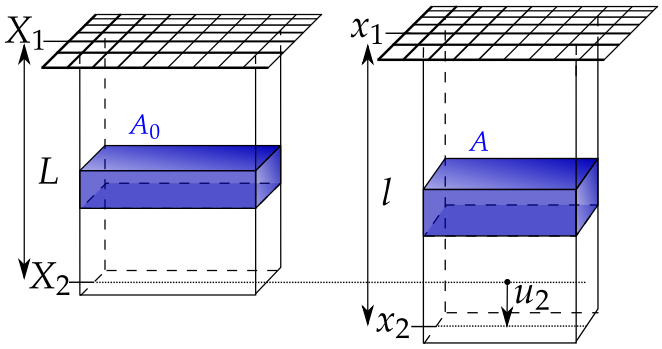


Figure 3.5: Reference and deformed configuration

3.1.2.2 Nonlinear functional

In this section we use engineering notation rather than the previously used mathematical notation. This is done in order to keep the equations short. It is obvious that δu_i is not the test function. By comparing with equation (3.10) on page 9 and equation (3.11) on page 9, it is easy to show that this is not the test function $w(x)$. It is rather a scalar, which is multiplied by the basis for the trial space $w^h(x) = \sum_{i=1}^{N} \phi_i(x)\delta u_i$. As δu_i is arbitrary, it is not stated in equation (3.11) on page 9. Note also that the superscript h is dropped in the following to simplify the notation.

We start with the quadratic energy functional for the hanging truss which reads:

$$\Pi = \frac{1}{2}\int_V \left[E\epsilon^2 + u(x)n\right] \mathrm{d}V \tag{3.28}$$

the weak form of equation (3.28) reads:

$$\delta\Pi = \int_V \left[E\epsilon(u(x))\delta\epsilon(u(x)) + \delta u(x)n\right] \mathrm{d}V = 0 \tag{3.29}$$

In the following we exploit the fact, that $u_1 = 0\,\mathrm{m}$ (because of the boundary condition), thus we can interpolate u by:

$$u(x) = \sum_{i=2}^{N} \phi_i(x)u_i = (1-x)u_2 \tag{3.30}$$

With this linear interpolation the functional can be written for the Hencky strain measure as

$$\Pi = \frac{1}{2}EA_0 \int_L \left[\left(\epsilon^{ln}\right)^2 + (1-x)u_2 n\right] \mathrm{d}L \tag{3.31}$$

The weak form of equation (3.29) on the previous page now reads:

$$\delta\Pi = EA_0 \int_L \left[\epsilon^{ln}\delta\epsilon^{ln}\right] \mathrm{d}L - \frac{1}{4}\delta u_2 = 0 \tag{3.32}$$

In order to be able to solve for u_2, we need ϵ^{ln} and $\delta\epsilon^{ln}$ depending on u_2

$$\epsilon^{ln} = ln\left(\frac{l}{L}\right) = ln\left(\frac{x_2 - x_1}{X_2 - X_1}\right) = ln\left(\frac{x_2 - 0}{1 - 0}\right) = ln\left(X_2 + u_2\right) = ln\left(1 + u_2\right) \tag{3.33}$$

Alternatively one can derive equation (3.33) by:

$$\epsilon^{ln} = ln\left(1 + \epsilon^{eng}\right) = ln\left(1 + \frac{\mathrm{d}u(x)}{\mathrm{d}x}\right) = ln\left(1 + u_2\right) \tag{3.34}$$

For the variation $\delta\epsilon^{ln}$ one has

$$\delta\epsilon^{ln} = \left(\frac{1}{1 + u_2}\right)\delta u_2 \tag{3.35}$$

Thus we can discretize the weak form (equation (3.29) on the previous page). The Galerkin form of the problem reads now:

$$\delta\Pi = \underbrace{EA_0\left(\frac{1}{1 + u_2} ln\left(1 + u_2\right)\right)}_{\mathbf{p}\to p \text{ internal forces}}\delta u_2 - \underbrace{0.25}_{\mathbf{f}\to f \text{ external forces}}\delta u_2 = 0 \tag{3.36}$$

This is not longer linear in u_2, hence one cannot separate the displacements and solve a linear equation: An iterative solution method is needed to solve the non-linear equation. Most often the Newton-Raphson algorithm is applied, which has quadratic convergence within the convergence radius. The Newton's method (see 3.1.2.3) needs the first derivative of the nonlinear equation (system). This is called tangent stiffness matrix and abbreviated by $\mathbf{K}_T$. Note that in the linear case the stiffness matrix coincides with the tangent stiffness matrix and the problem is solved with one "iteration" (linear equation system).

$$\mathbf{K}_T \to K_T = \frac{\partial p}{\partial u_2} = \frac{1}{(1 + u_2)^2} - \frac{ln(1 + u_2)}{(1 + u2)^2} \tag{3.37}$$

We also need the residual of the weak form of the problem (equation (3.36)) for the Newton-Raphson algorithm, which is defined by the difference between internal and external forces, hence by:

$$\mathbf{r} \to r = p - f = \frac{1}{1 + u_2} ln\left(1 + u_2\right) - 0.25 \tag{3.38}$$

It is now possible to define the equilibrium path, which is a function of u_2. This reads for equation (3.36) on the previous page:

$$f = \frac{1}{1+u_2} ln\,(1+u_2) \tag{3.39}$$

Applying the same procedure from above for the Green-Lagrange strain measure, one can also arrive at the equilibrium path for the Green-Lagrange strain:

$$f = (1+u_2)\left(u_2 + \frac{1}{2}u_2^2\right) \tag{3.40}$$

If we plot the equilibrium paths for the Hencky strain (figure 3.6) and for the Green-Lagrange strain (figure 3.7 on the next page) and compare them (figure 3.8 on the following page), one can directly see that for the Hencky strain and an external load of $f = 0.5\,\mathrm{N}$ there is no solution within $\mathbb{R}$. Therefore, the example was modified to an external load of $f = 0.25\,\mathrm{N}$.

It can also be derived, that the Green-Lagrange strain in combination with this simple material model does not coincide with physics for large strains. Because of the assumption that the volume is constant, the stiffness of the bar should decrease during the deformation process. But the Green-Lagrange strain gives a smaller value than the linear solution $u_2^{GL} = 0.1914\,\mathrm{m} < u_2^{linear} = 0.2500\,\mathrm{m}$. The Hencky strain on the other side coincides with physical expectations $u_2^{ln} = 0.4296\,\mathrm{m} > u_2^{linear} = 0.2500\,\mathrm{m}$

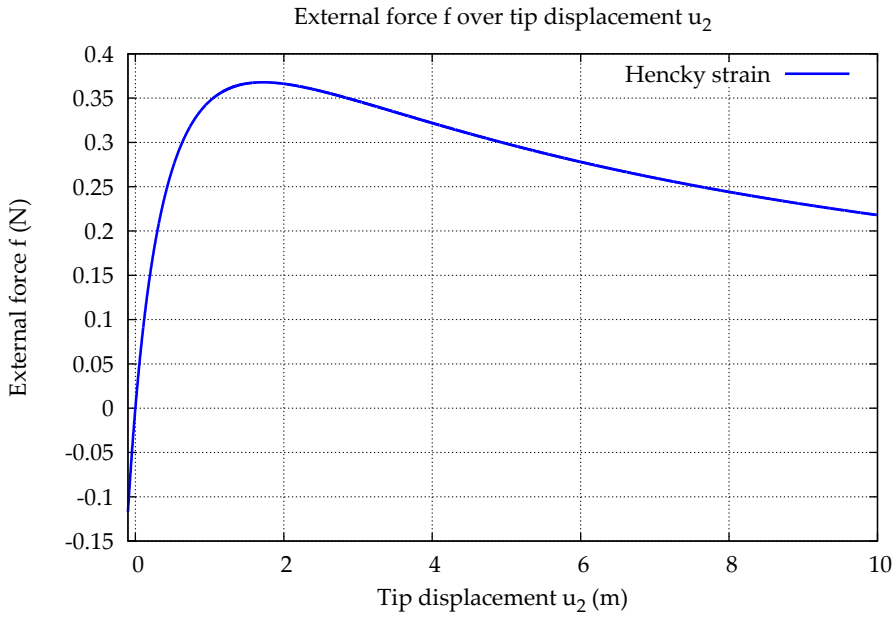


Figure 3.6: Equilibrium path for the hanging truss with Hencky strain measure

3.1.2.3 Newton-Raphson algorithm

In order to solve the nonlinear equation (3.36) on the previous page, an iterative technique is applied. The so-called Newton-Raphson algorithm (see algorithm

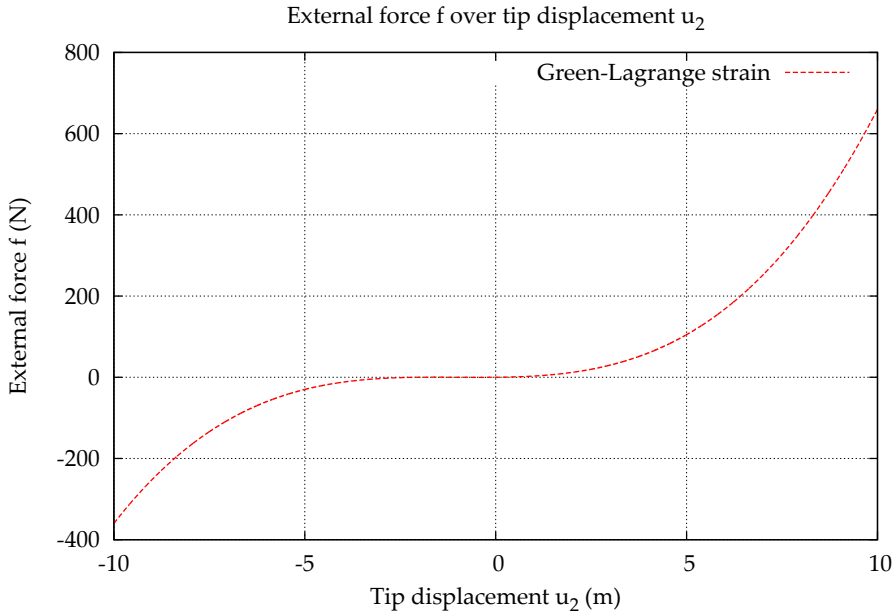


Figure 3.7: Equilibrium path for the hanging truss with Green-Lagrange strain measure

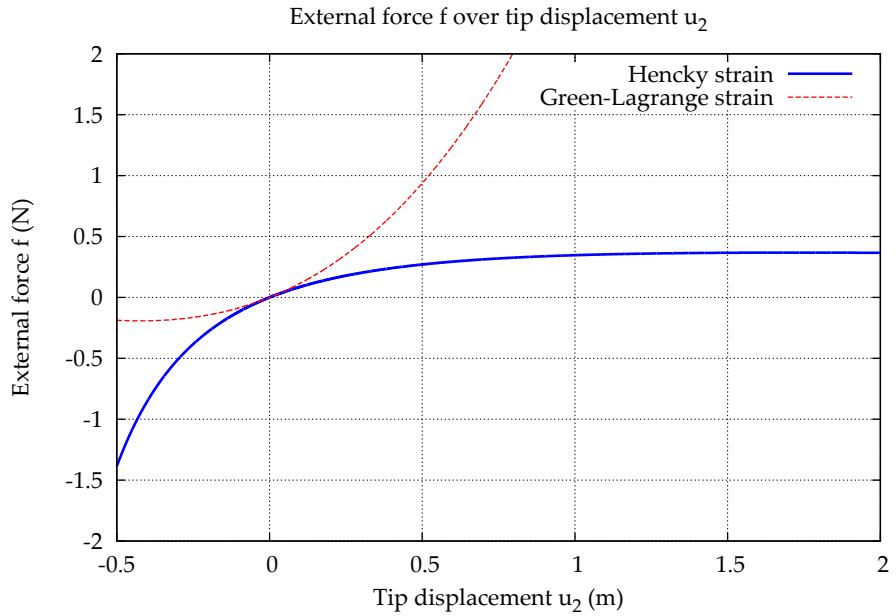


Figure 3.8: Equilibrium path for the hanging truss (closeup)

1) has the nice property of a quadratic convergence rate, if the distance between starting value $u_2^{(0)}$ and the solution u_2 is sufficiently small (see [24, page 171]). The algorithm needs the first derivative (tangent stiffness) of the weak form in order to achieve a quadratic convergence rate. Now we would like to investigate a little bit the nonlinear hanging truss problem for the Hencky strain measure. For a starting value of $u_2^{(0)} = 0\,\mathrm{m}$ (the iterations are illustrated in figure 3.9 on the following page), the corresponding values are stated in table 3.1 on page 19. From the table we can see that the quadratic convergence rate is achieved from

Algorithm 1 The Newton-Raphson algorithm

1: $i \leftarrow 0$
2: $\mathbf{u}^{(i)} \leftarrow \mathbf{u}^{(start)}$ ▷ Initialize with start vector
3: **while** $\left\| r(\mathbf{u}^{(i)}) \right\| > TOL$ **do** ▷ Check if converged
4: COMPUTE $\mathbf{r}(\mathbf{u}^{(i)})$
5: COMPUTE $\mathbf{K}_T(\mathbf{u}^{(i)})$
6: SOLVELINEARSYSTEM($\mathbf{K}_T(\mathbf{u}^{(i)})\Delta\mathbf{u}^{(i)} = -r(\mathbf{u}^{(i)})$) ▷ Solve for $\Delta\mathbf{u}^{(i)}$
7: UPDATEDISPLACMENTS($\mathbf{u}^{(i+1)} = \mathbf{u}^{(i)} + \Delta\mathbf{u}^{(i)}$)
8: **end while**
9: **return** $\mathbf{u}^{(i+1)}$ ▷ Return solution if converged

the second iteration on. That indicates that the starting value is a little bit too far away from the solution.

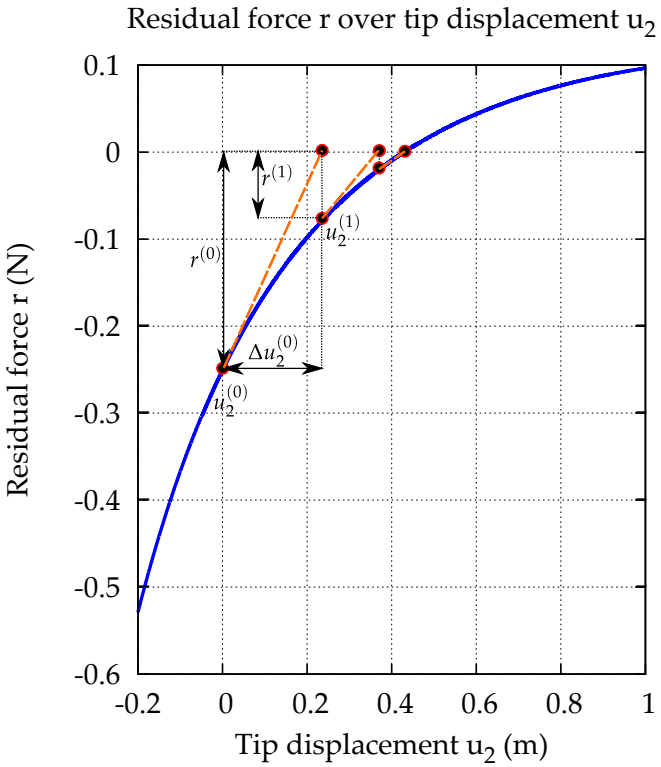


Figure 3.9: Illustration of Newton's method for an external load of $f = 0.25\,\mathrm{N}$ with $u_2^{(0)} = 0\,\mathrm{m}$

In contrast to the linear case, nonlinear problems do in general not have a unique solution. For the hanging truss this can be easily seen if we choose $u_2^{(0)} = 2\,\mathrm{m}$. Figure 3.10 on the following page demonstrates, where the Newton-Raphson algorithm will arrive for that starting value. Obviously the solution is different now to the one from figure 3.9. Both solutions are mathematically correct, nevertheless only the first one is mechanically correct.

Table 3.1: Newton iterations for an external load of $f = 0.25\,\mathrm{N}$

iteration	residual	u_2	abs. error in u_2	Δu_2
	N	m	m	m
0	−0.2500000000	0.0000000000	0.4296118247	0.2500000000
1	−0.0714851590	0.2500000000	0.1796118247	0.1437788938
2	−0.0117852526	0.3937788939	0.0358329308	0.0342738075
3	−0.0004911629	0.4280527011	0.0015591236	0.0015561032
4	−0.0000009498	0.4296088045	0.0000030202	0.0000030202
5	−0.0000000000	0.4296118247	0.0000000000	0.0000000000

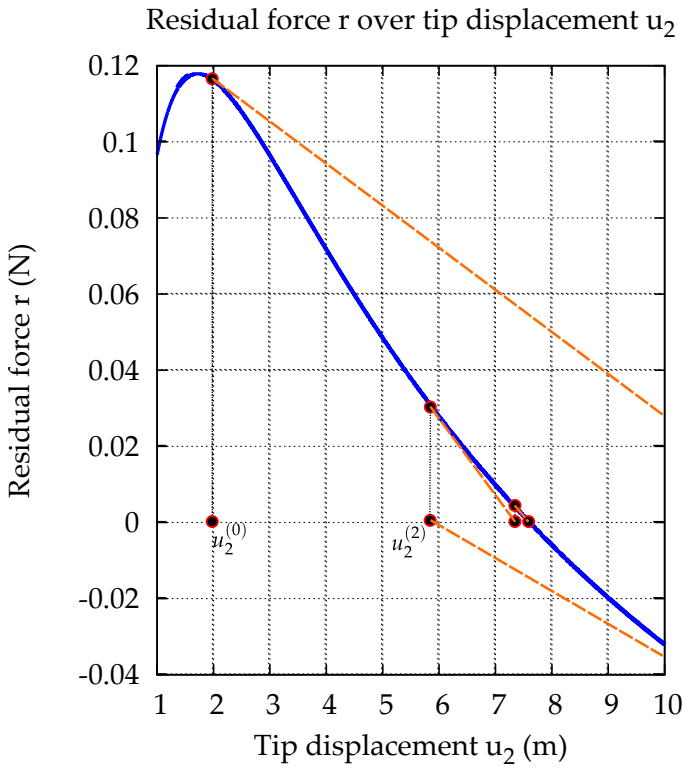


Figure 3.10: Illustration of Newton's method for an external load of $f = 0.25\,\mathrm{N}$ with $u_2^{(0)} = 2\,\mathrm{m}$

3.2 Contact mechanics and the penalty method

Contact is a nonlinear boundary condition. It can be formulated as a constrained functional. In this section we again stick to the hanging truss example. The two versions of the example are used again for linear and nonlinear implementations, respectively. Note that only the penalty method is discussed for constraint enforcement, although there are more methods available, like e.g. Lagrange multipliers. The references [31] and [16] give more insight into this topic. The penalty method cannot enforce the constraint exactly. There is always a slight penetration. Just mathematically it would be possible, to enforce the constraint exactly by an

infinite large penalty stiffness c. On the other hand Lagrange multiplier methods can enforce the constraint exactly, but they are more complex to implement and show in general a worse convergence behavior.

3.2.1 Linear contact kinematics

The modified functional can be written as

$$\widetilde{\Pi} = \overbrace{\underbrace{\frac{1}{2}\int_V \left[E(u'(x))^2 + u(x)n\right] \mathrm{d}V}_{\text{structural contribution}}}^{\Pi} + \overbrace{\underbrace{\frac{1}{2}\int_{\partial V} cg^2 \mathrm{d}\partial V}_{\text{penalty contact contribution}}}^{\Pi^C_{Penalty}} \tag{3.41}$$

Since the hanging truss is a one-dimensional linear example, we can reformulate the gap function (see equation (4.2) on page 24) to:

$$g = (u_5 - u_4) + g_{ini} \tag{3.42}$$

Now the penalty part of the functional reads:

$$\Pi^C_{Penalty} = \frac{1}{2}\int_{\partial V} \left[c((u_5 - u_4) + g_{ini})^2\right] \mathrm{d}\partial V \tag{3.43}$$

By applying the rules of variational calculus the first variation of the problem reads:

$$\delta\Pi^C_{Penalty} = c\int_{\partial V} \left[(u_5 - u_4) + g_{ini}(\delta u_5 - \delta u_4)\right] \mathrm{d}\partial V \tag{3.44}$$

Equation (3.44) has the form $\delta\Pi = \frac{\partial\Pi}{\partial u_4}\delta u_4 + \frac{\partial\Pi}{\partial u_5}\delta u_5$ in order to render the functional to stationary point the partial derivatives (residual) must become zero as it must hold for arbitrary virtual displacements (test function scalars). We can reformulate the weak form now in a linear system of equations. The integral over the

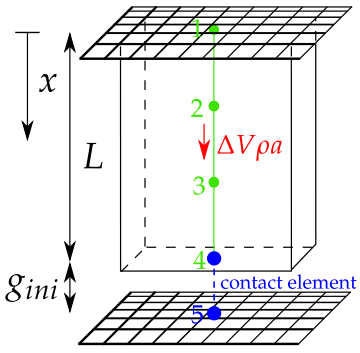


Figure 3.11: Hanging truss subjected to gravity with contact

body boundaries ∂V can be replaced by a multiplication by the truss cross section A_0 (contact surface), as only one node can come into contact.

$$\mathbf{K}^C \mathbf{u}^C = \mathbf{f}^C \tag{3.45}$$

$$A_0 c \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} u_4 \\ u_5 \end{bmatrix} = c \begin{bmatrix} g_{ini} \\ -g_{ini} \end{bmatrix} \tag{3.46}$$

The assembled linear equation system for the hanging truss reads now for an active contact constraint:

$$\begin{bmatrix} 6 & -3 & 0 \\ -3 & 6 & -3 \\ 0 & -3 & 3+c \end{bmatrix} \begin{bmatrix} u_2 \\ u_3 \\ u_4 \end{bmatrix} = \begin{bmatrix} \frac{1}{3} \\ \frac{1}{3} \\ \frac{1}{6} + g_{ini}\, c \end{bmatrix} \tag{3.47}$$

The solution for an initial gap of $g_{ini} = 0.1\,\mathrm{m}$ is plotted in figure 3.12.

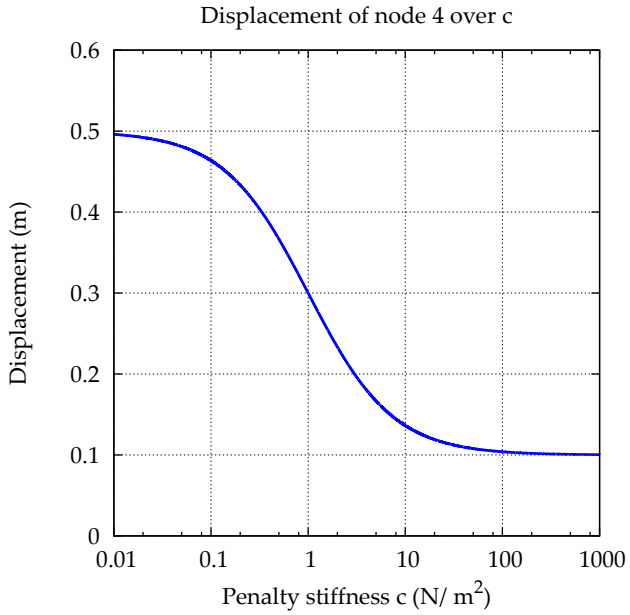


Figure 3.12: Influence of the penalty stiffness c for linear contact kinematics

3.2.2 Nonlinear contact kinematics

As we now derive a nonlinear contact element, it makes sense to use it in combination with a nonlinear truss element, even if a combination with a linear truss element would also be possible. The functional for the nonlinear hanging truss example including contact is given by:

$$\widetilde{\Pi} = \frac{1}{2} \int_V \left[E(\epsilon^{ln})^2 + u(x) n \right] \mathrm{d}V + \underbrace{\frac{1}{2} \int_{\partial V} \left[c g^2 \right] \mathrm{d}\partial V}_{\text{penalty contact contribution}} \tag{3.48}$$

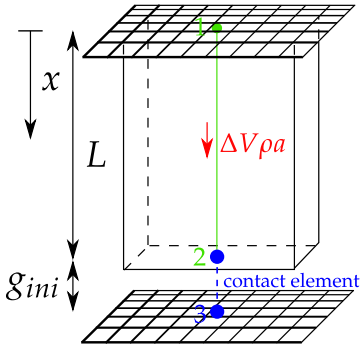


Figure 3.13: Hanging truss subjected to gravity with contact (nonlinear formulation)

As we will check the contact status in each iteration the gap function simplifies to

$$g = u_3 - u_2 \tag{3.49}$$

Now the penalty part of the functional reads

$$\Pi^C_{Penalty} = \frac{1}{2} \int_{\partial V} \left[c(u_3 - u_2)^2 \right] \mathrm{d}\partial V \tag{3.50}$$

By applying the rules of variational calculus the first variation of the problem reads

$$\delta\Pi^C_{Penalty} = c \int_{\partial V} \left[(u_3 - u_2)(\delta u_3 - \delta u_2) \right] \mathrm{d}\partial V \tag{3.51}$$

The contact contribution to the global residual vector is

$$\mathbf{r}^C = cg \begin{bmatrix} -1 \\ 1 \end{bmatrix} \tag{3.52}$$

From equation (3.51) we can derive the tangent stiffness matrix of the contact element. Note that the tangent stiffness matrix of the contact element does not depend on $\mathbf{u}^C$. This is because of the one-dimensional case, for which the normal vector does not change (see chapter 4 for further explanation).

$$\mathbf{K}^C_T = c \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \tag{3.53}$$

For the nonlinear case special solution techniques are required. Thus, we investigate a basic contact algorithm which can solve the hanging truss problem (see algorithm 2). The critical line in the contact algorithm 2 is line number 8. It determines, if contact is active or not. If the penalty method is used, we always have slight penetrations. This means that the gap is negative for a closed (active) contact pair. Because we have always this slight penetration it is not necessary

Algorithm 2 Contact solution algorithm for the penalty method for small sliding

```
CONTACTSEARCH                                              ▷ Get potential contact pairs
i ← 0
u^(i) ← u^(start)                                          ▷ Initialize with start vector
while ||r(u^(i))|| > TOL do                                ▷ Check if converged
    EVALUTEGAPFUNCTION(u^C(i))                             ▷ For all potential contact pairs
    COMPUTE K_T(u^(i))                                     ▷ Global tangent stiffness
    COMPUTE r(u^(i))                                       ▷ Global residual vector
    if g(u^C(i)) < 0 then                                  ▷ Contact status is active
        COMPUTE K_T^C(u^C(i))                              ▷ Contact tangent stiffness
        COMPUTE r^C(u^C(i))                                ▷ Contact residual vector
        ADDCONTACTSTIFFNESS(K_T^C(u^C(i)))                 ▷ Add to K_T
        ADDCONTACTRESIDUAL(r^C(u^C(i)))                    ▷ Add to r
    end if
    SOLVELINEARSYSTEM(K_T(u^(i))Δu^(i) = −r(u^(i)))        ▷ Solve for Δu^(i)
    UPDATEDISPLACMENTS(u^(i+1) = u^(i) + Δu^(i))
end while
return u^(i+1)                                             ▷ Return solution if converged
```

to check whether the contact force is a pressure force. A tension would create a positive gap, resulting in a non-active contact pair.

The tangent stiffness matrix and residual vector for an active contact status reads for the hanging truss example

$$r = p - f = \frac{1}{1+u_2} ln\,(1+u_2) - cg - 0.25 \tag{3.54}$$

$$K_T = \frac{1}{(1+u_2)^2} - \frac{ln(1+u_2)}{(1+u_2)^2} + c \tag{3.55}$$

Table 3.2 shows the results for algorithm 2 in detail for a penalty stiffness of $c = 1000\,\mathrm{N/m^2}$

Table 3.2: Newton iterations for an external load of $f = 0.25\,\mathrm{N}$ and $c = 1000\,\mathrm{N/m^2}$

iteration	residual	$u_2 + \Delta u_2$	gap	contact status
	N	m	m	
0	−0.2500000000	0.2500000000	0.1000000000	open
1	149.9285148411	0.1001459908	−0.1500000000	close
2	−0.0172544573	0.1001632324	−0.0001459908	close
3	−0.0000000003	0.1001632324	−0.0001632324	close

Chapter 4

The 4-node contact element

In this section the concepts of chapter 3 are generalized to three dimensions to derive the implemented contact element. It is a node-to-surface formulation in combination with a penalty method for enforcing the normal constraint. No tangential effects (like friction) are taken into account. The focus in the following lies on a 4-node contact element. Which means that the master surface is built from triangular faces. A triangle has the advantage (in contrast to a bilinear quadrilateral) that the normal is constant within a face. No contact smoothing is applied, hence the normal can jump between neighboring triangles.

4.1 Contact kinematics

In this section an overview of the nonlinear contact kinematics is given. The basic ideas are always kept as general as possible. The final equations, which are needed for an implementation, are derived for the 4-node contact element only.

4.1.1 Gap function

Assume that two bodies come into contact. In this case the penetration function can be written as:

$$g = (\mathbf{x}^S - \bar{\mathbf{x}}^M) \circ \bar{\mathbf{n}}^M < 0 \tag{4.1}$$

In figure 4.1 on the next page the problem of the two body contact is illustrated. It is assumed that the two contact surfaces are at least locally convex. The gap function (in the penalty case also called penetration function) is stated as:

$$g = (\mathbf{x}^S - \bar{\mathbf{x}}^M) \circ \bar{\mathbf{n}}^M \tag{4.2}$$

Where the slave node (indicated by the superscript s) is projected onto the master surface, which results in the projection point $\bar{\mathbf{x}}^M$ the bar always indicates quantities evaluated at this projection point. The vectors $\bar{\mathbf{a}}_1^M$ and $\bar{\mathbf{a}}_2^M$ are the covariant tangent vectors. The unit normal vector $\bar{\mathbf{n}}^M$ is defined by:

$$\bar{\mathbf{n}}^M = \frac{\bar{\mathbf{a}}_1^M \times \bar{\mathbf{a}}_2^M}{\left\| \bar{\mathbf{a}}_1^M \times \bar{\mathbf{a}}_2^M \right\|} \tag{4.3}$$

The coordinates ξ_1, ξ_2 are called convective coordinates. They are engraved in the material, therefore they deform with the material.

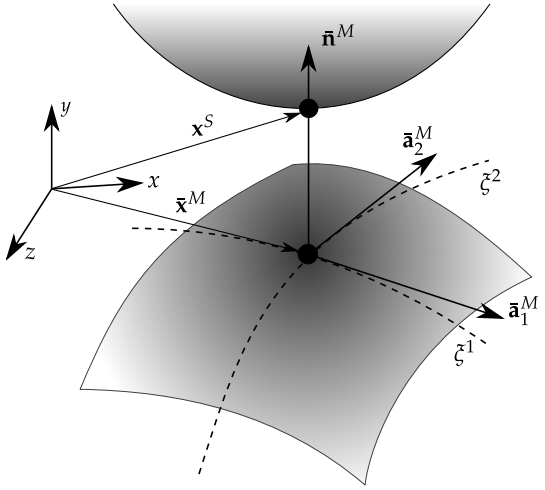


Figure 4.1: Gap function illustration

4.1.2 The projection problem

A question which may arise immediately while reading the previous section is how to compute the slave projection point $\bar{\mathbf{x}}^M$. The basic idea is to minimize the distance b between the slave node and the master segment. This can be expressed by:

$$b(\xi_1, \xi_2) = \left\| \mathbf{x}^S - \mathbf{x}^M(\bar{\xi}_1, \bar{\xi}_2) \right\| = min \left\| \mathbf{x}^S - \mathbf{x}^M(\xi_1, \xi_2) \right\| \tag{4.4}$$

The solution to this problem can be found by setting the first derivative to zero due to the assumption that we have local convexity of the contact bodies.

$$\frac{\mathrm{d}b(\xi_1, \xi_2)}{\mathrm{d}\xi_\alpha} = \frac{\mathbf{x}^S - \mathbf{x}^M(\xi_1, \xi_2)}{\|\mathbf{x}^S - \mathbf{x}^M(\xi_1, \xi_2)\|} \circ \frac{\mathrm{d}\mathbf{x}^M(\xi_1, \xi_2)}{\mathrm{d}\xi_\alpha} = 0 \tag{4.5}$$

This can be simplified by using equation (4.7) to

$$\mathbf{x}^S - \mathbf{x}^M(\xi_1, \xi_2) \circ \frac{\mathrm{d}\mathbf{x}^M(\xi_1, \xi_2)}{\mathrm{d}\xi_\alpha} = \mathbf{x}^S - \mathbf{x}^M(\xi_1, \xi_2) \circ \mathbf{a}_\alpha^M = 0 \tag{4.6}$$

$$\frac{\mathrm{d}\mathbf{x}^M(\xi_1, \xi_2)}{\mathrm{d}\xi_\alpha} = \mathbf{a}_\alpha^M \tag{4.7}$$

For the 4-node contact element (see figure 4.2 on the next page) the equation set (4.6) is linear. According to figure 4.2 on the following page it is also possible to

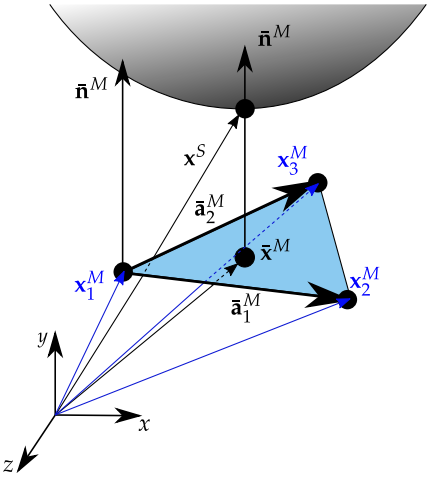


Figure 4.2: Gap function illustration for a 4-node contact element

define the tangent vectors as

$$\bar{\mathbf{a}}_1^M = \mathbf{x}_2^M - \mathbf{x}_1^M \tag{4.8}$$
$$\bar{\mathbf{a}}_2^M = \mathbf{x}_3^M - \mathbf{x}_1^M \tag{4.9}$$

Because the tangent vectors are constant within the master face equation (4.2) on page 24 is equivalent to

$$g = (\mathbf{x}^S - \mathbf{x}^M) \circ \mathbf{n}^M < 0 \tag{4.10}$$

The shape functions for a linear triangle may be given by:

$$N_1(\xi_1, \xi_2) = 1 - \xi_1 - \xi_2 \tag{4.11}$$
$$N_2(\xi_1, \xi_2) = \xi_1 \tag{4.12}$$
$$N_3(\xi_1, \xi_2) = \xi_2 \tag{4.13}$$

By using the isoparametric concept it is possible to interpolate a node on the master surface by:

$$\mathbf{x}^M(\xi_1, \xi_2) = \sum_{i=1}^{3} N_i(\xi_1, \xi_2)\mathbf{x}_i^M \tag{4.14}$$

Now equation (4.6) on the preceding page reads:

$$\left[\mathbf{x}^S - \sum_{i=1}^{3} N_i(\bar{\xi}_1, \bar{\xi}_2)\mathbf{x}_i^M\right] \circ \bar{\mathbf{a}}_\alpha^M = 0 \tag{4.15}$$

$$\begin{bmatrix} \bar{\mathbf{a}}_1^M \circ \bar{\mathbf{a}}_1^M & \bar{\mathbf{a}}_1^M \circ \bar{\mathbf{a}}_2^M \\ \bar{\mathbf{a}}_1^M \circ \bar{\mathbf{a}}_2^M & \bar{\mathbf{a}}_2^M \circ \bar{\mathbf{a}}_2^M \end{bmatrix} \begin{bmatrix} \bar{\xi}_1 \\ \bar{\xi}_2 \end{bmatrix} = \begin{bmatrix} (\mathbf{x}^S - \mathbf{x}_1^M) \circ \bar{\mathbf{a}}_1^M \\ (\mathbf{x}^S - \mathbf{x}_1^M) \circ \bar{\mathbf{a}}_2^M \end{bmatrix} \tag{4.16}$$

Now it is fairly easy possible to compute $\bar{\xi}_1$ and $\bar{\xi}_2$, hence one can compute the projection point by equation (4.14).

4.2 Contact discretization

First we should recall equation (3.41) on page 20 because we will start from the same functional now. The penalty functional reads:

$$\Pi^{C}_{Penalty} = \frac{1}{2}\int_{\partial V} cg^2 \mathrm{d}\partial V \tag{4.17}$$

In the following we need to derive a tangent stiffness matrix and a residual force vector from the functional. If we perform the first variation of equation (4.17) we arrive at

$$\delta\Pi^{C}_{Penalty} = c\int_{\partial V} g\delta g \mathrm{d}\partial V \tag{4.18}$$

This integral is discretized by the node-to-surface formulation to a sum

$$\delta\Pi^{C}_{Penalty} = c\int_{\partial V} g\delta g \mathrm{d}\partial V \rightarrow c\sum_{n=1}^{n_c} A_n g_n \delta g_n \tag{4.19}$$

where A_n is the contact surface area which is connected to the slave node n. And n_c is the number of slave nodes used for the discretization of the contact zone.

4.2.1 Variation of the contact penalty functional

Equation (4.19) is already the first variation of the contact contribution. In order to be able to derive the residual vector from it, δg needs to be expressed in terms of $\mathbf{x}_i^M$ and $\mathbf{x}^S$. Note that for a triangular master face the normal and hence the tangent vectors are constant within each face. Note also that $\mathbf{x}_i^M = \mathbf{X}_i^M + \mathbf{u}_i^M$ and $\mathbf{x}^S = \mathbf{X}^S + \mathbf{u}^S$. The derivation of δg is expressed in equation (4.24) on the next page.

4.2.2 Linearization of the variation of the contact penalty functional

The directional derivative (see appendix A.1) is denoted by Δ in the following. Keep in mind, that a linearization in terms of the displacements $\mathbf{u}_i^M$ and $\mathbf{u}^S$ is needed. But it would make things to complicated to start right away with the displacements. Hence, the linearization is derived in step by step manner.

$$\Delta\delta\Pi^{C}_{Penalty} = c\sum_{n=1}^{n_c} A_n(\Delta g_n \delta g_n + g_n \Delta\delta g_n) \tag{4.20}$$

4.2.2.1 Derivation of involved terms

We will derive terms for Δg, δg and $\Delta\delta g$ in the following paragraphs. After that we are able to come up with a solution to equation (4.20).

4.2.2.1.1 The term Δg We start by applying the generalized chain rule and consider every possible deformation dependence, which yields to

$$\Delta g = \left[\Delta \mathbf{x}^S - \Delta \bar{\mathbf{x}}^M - \frac{\mathrm{d}\mathbf{x}^M(\bar{\xi}_1, \bar{\xi}_2)}{\mathrm{d}\bar{\xi}_\alpha} \Delta \bar{\xi}_\alpha\right] \circ \bar{\mathbf{n}}^M + \left[\mathbf{x}^S - \bar{\mathbf{x}}^M\right] \circ \Delta \bar{\mathbf{n}}^M \tag{4.21}$$

Equation (4.21) can be considerably reduced by knowing, that $\bar{\mathbf{n}}^M \circ \Delta \bar{\mathbf{n}}^M = 0$ and $\bar{\mathbf{a}}_\alpha^M \circ \bar{\mathbf{n}}^M = 0$. With the help of appendix A.1 we can derive Δg now step by step.

$$\begin{aligned}
\Delta g &= \Delta \left[\mathbf{x}^S - \bar{\mathbf{x}}^M\right] \circ \bar{\mathbf{n}}^M = \Delta \left[\mathbf{x}^S - \sum_{i=1}^{3} N_i(\bar{\xi}_1, \bar{\xi}_2) \mathbf{x}_i^M\right] \circ \bar{\mathbf{n}}^M \qquad (4.22)\\
&= \Delta \left[\mathbf{x}^S - ((1 - \bar{\xi}_1 - \bar{\xi}_2)\mathbf{x}_1^M) + (\bar{\xi}_1 \mathbf{x}_2^M) + (\bar{\xi}_2 \mathbf{x}_3^M)\right] \circ \bar{\mathbf{n}}^M \\
&= \begin{bmatrix}
\frac{\partial\left[(\mathbf{x}^S - ((1-\bar{\xi}_1-\bar{\xi}_2)\mathbf{x}_1^M) + (\bar{\xi}_1\mathbf{x}_2^M) + (\bar{\xi}_2\mathbf{x}_3^M)) \circ \bar{\mathbf{n}}^M\right]}{\partial \mathbf{x}^S} \\
\frac{\partial\left[(\mathbf{x}^S - ((1-\bar{\xi}_1-\bar{\xi}_2)\mathbf{x}_1^M) + (\bar{\xi}_1\mathbf{x}_2^M) + (\bar{\xi}_2\mathbf{x}_3^M)) \circ \bar{\mathbf{n}}^M\right]}{\partial \mathbf{x}_1^M} \\
\frac{\partial\left[(\mathbf{x}^S - ((1-\bar{\xi}_1-\bar{\xi}_2)\mathbf{x}_1^M) + (\bar{\xi}_1\mathbf{x}_2^M) + (\bar{\xi}_2\mathbf{x}_3^M)) \circ \bar{\mathbf{n}}^M\right]}{\partial \mathbf{x}_2^M} \\
\frac{\partial\left[(\mathbf{x}^S - ((1-\bar{\xi}_1-\bar{\xi}_2)\mathbf{x}_1^M) + (\bar{\xi}_1\mathbf{x}_2^M) + (\bar{\xi}_2\mathbf{x}_3^M)) \circ \bar{\mathbf{n}}^M\right]}{\partial \mathbf{x}_3^M}
\end{bmatrix} \circ \begin{bmatrix} \Delta \mathbf{u}^S \\ \Delta \mathbf{u}_1^M \\ \Delta \mathbf{u}_2^M \\ \Delta \mathbf{u}_3^M \end{bmatrix} \\
&= \begin{bmatrix} \bar{\mathbf{n}}^M \\ -(1 - \bar{\xi}_1 - \bar{\xi}_2)\bar{\mathbf{n}}^M \\ -(\bar{\xi}_1)\bar{\mathbf{n}}^M \\ -(\bar{\xi}_2)\bar{\mathbf{n}}^M \end{bmatrix} \circ \begin{bmatrix} \Delta \mathbf{u}^S \\ \Delta \mathbf{u}_1^M \\ \Delta \mathbf{u}_2^M \\ \Delta \mathbf{u}_3^M \end{bmatrix} \\
&= (\Delta \mathbf{u}^S - \Delta \bar{\mathbf{u}}^M) \circ \bar{\mathbf{n}}^M \qquad (4.23)
\end{aligned}$$

4.2.2.1.2 The term δg The same mechanisms from the previous paragraph can be applied to derive an equation for δg.

$$\delta g = (\delta \mathbf{x}^S - \delta \bar{\mathbf{x}}^M) \circ \bar{\mathbf{n}}^M = \left(\delta \mathbf{x}^S - \sum_{i=1}^{3} N_i(\bar{\xi}_1, \bar{\xi}_2) \delta \mathbf{x}_i^M\right) \circ \bar{\mathbf{n}}^M \tag{4.24}$$

4.2.2.1.3 The term $\Delta \delta g$ Only $\Delta \delta g$ is missing in order to be able to formulate the residual vector and tangent stiffness matrix of the 4-node contact element. But $\Delta \delta g$ is more complicated to derive. We need to start from equation (4.2) on page 24

$$g = (\mathbf{x}^S - \bar{\mathbf{x}}^M) \circ \bar{\mathbf{n}}^M \tag{4.25}$$

$$g \bar{\mathbf{n}}^M = (\mathbf{x}^S - \bar{\mathbf{x}}^M) \tag{4.26}$$

the next step is the variation

$$\delta g \bar{\mathbf{n}}^M + g \delta \bar{\mathbf{n}}^M = \delta \mathbf{x}^S - \delta \bar{\mathbf{x}}^M - \bar{\mathbf{x}}_{,\alpha}^M \delta \bar{\xi}_\alpha \tag{4.27}$$

followed by the linearization of the variation (second order derivatives are neglected)

$$\Delta\delta g\bar{\mathbf{n}}^M + \delta g\Delta\bar{\mathbf{n}}^M + \Delta g\delta\bar{\mathbf{n}}^M + g\Delta\delta\bar{\mathbf{n}}^M = -\delta\bar{\mathbf{x}}^M_{,\alpha}\Delta\bar{\xi}_\alpha - \Delta\bar{\mathbf{u}}^M_{,\alpha}\delta\bar{\xi}_\alpha - \bar{\mathbf{x}}^M_{,\alpha}\Delta\delta\bar{\xi}_\alpha \quad (4.28)$$

$$\Delta\delta g + g\Delta\delta\bar{\mathbf{n}}^M \circ \bar{\mathbf{n}}^M = (-\delta\bar{\mathbf{x}}^M_{,\alpha}\Delta\bar{\xi}_\alpha - \Delta\bar{\mathbf{u}}^M_{,\alpha}\delta\bar{\xi}_\alpha - \bar{\mathbf{x}}^M_{,\alpha}\Delta\delta\bar{\xi}_\alpha) \circ \bar{\mathbf{n}}^M \quad (4.29)$$

$$\Delta\delta g = (-\delta\bar{\mathbf{x}}^M_{,\alpha}\Delta\bar{\xi}_\alpha - \Delta\bar{\mathbf{u}}^M_{,\alpha}\delta\bar{\xi}_\alpha - \bar{\mathbf{x}}^M_{,\alpha}\Delta\delta\bar{\xi}_\alpha) \circ \bar{\mathbf{n}}^M - g\Delta\delta\bar{\mathbf{n}}^M \circ \bar{\mathbf{n}}^M \quad (4.30)$$

There is no need to compute $\Delta\delta\bar{\mathbf{n}}^M$, because $\Delta(\bar{\mathbf{n}}^M \circ \delta\bar{\mathbf{n}}^M) = \Delta\bar{\mathbf{n}}^M \circ \delta\bar{\mathbf{n}}^M + \bar{\mathbf{n}}^M \circ \Delta\delta\bar{\mathbf{n}}^M = 0$. Now we have $\Delta\bar{\mathbf{n}}^M \circ \delta\bar{\mathbf{n}}^M = -\bar{\mathbf{n}}^M \circ \Delta\delta\bar{\mathbf{n}}^M$. Now the linearized variation of the gap is given by:

$$\Delta\delta g = (-\delta\bar{\mathbf{x}}^M_{,\alpha}\Delta\bar{\xi}_\alpha - \Delta\bar{\mathbf{u}}^M_{,\alpha}\delta\bar{\xi}_\alpha) \circ \bar{\mathbf{n}}^M + g\delta\bar{\mathbf{n}}^M \circ \Delta\bar{\mathbf{n}}^M \quad (4.31)$$

In the following we will derive the last terms which are missing.

4.2.2.1.3.1 The term $\Delta\bar{\mathbf{n}}^M$ First we derive an expression for $\Delta\bar{\mathbf{n}}^M$

$$\bar{\mathbf{a}}^M_\alpha \circ \bar{\mathbf{n}}^M = 0 \rightarrow \Delta\bar{\mathbf{a}}^M_\alpha \circ \bar{\mathbf{n}}^M + \bar{\mathbf{a}}^M_\alpha \circ \Delta\bar{\mathbf{n}}^M = 0 \quad (4.32)$$

$$\begin{aligned} \bar{\mathbf{a}}^M_\alpha \circ \Delta\bar{\mathbf{n}}^M &= -\Delta\bar{\mathbf{a}}^M_\alpha \circ \bar{\mathbf{n}}^M \\ (\Delta\bar{\mathbf{n}}^M \circ \bar{\mathbf{a}}^M_\alpha)\bar{\mathbf{a}}^{M\alpha} &= -(\bar{\mathbf{n}}^M \circ \Delta\bar{\mathbf{a}}^M_\alpha)\bar{\mathbf{a}}^{M\alpha} \\ \Delta\bar{\mathbf{n}}^M &= -(\bar{\mathbf{n}}^M \circ \Delta\bar{\mathbf{a}}^M_\alpha)\bar{\mathbf{a}}^{M\alpha} \\ \Delta\bar{\mathbf{n}}^M &= -(\bar{\mathbf{n}}^M \circ \Delta\bar{\mathbf{a}}^M_\alpha)\bar{a}^{\alpha\beta}\bar{\mathbf{a}}^M_\beta \end{aligned} \quad (4.33)$$

An expression for $\Delta\bar{\mathbf{a}}^M_\alpha$ is derived next

$$\bar{\mathbf{a}}^M_\alpha = \frac{\mathrm{d}\mathbf{x}^M(\bar{\xi}_1, \bar{\xi}_2)}{\mathrm{d}\bar{\xi}_\alpha} = \mathrm{d}\mathbf{x}^M(\bar{\xi}_1, \bar{\xi}_2)_{,\alpha} = \begin{bmatrix} N_{1,\alpha} \\ N_{2,\alpha} \\ N_{3,\alpha} \end{bmatrix} \circ \begin{bmatrix} \mathbf{x}^M_1 \\ \mathbf{x}^M_2 \\ \mathbf{x}^M_3 \end{bmatrix} \quad (4.34)$$

Applying the directional derivative reads then

$$\Delta\bar{\mathbf{a}}^M_\alpha = \Delta\bar{\mathbf{u}}^M_{,\alpha} \quad (4.35)$$

Finally we have

$$\Delta\bar{\mathbf{n}}^M = -(\bar{\mathbf{n}}^M \circ \Delta\bar{\mathbf{u}}^M_{,\alpha})\bar{a}^{\alpha\beta}\bar{\mathbf{a}}^M_\beta \quad (4.36)$$

4.2.2.1.3.2 The term $\delta\bar{\mathbf{n}}^M$ From the previous equation we can directly state an expression for

$$\delta\bar{\mathbf{n}}^M = -(\bar{\mathbf{n}}^M \circ \delta\bar{\mathbf{x}}^M_{,\alpha})\bar{a}^{\alpha\beta}\bar{\mathbf{a}}^M_\beta \quad (4.37)$$

4.2.2.1.3.3 The term $\Delta\bar{\xi}_\alpha$

This time we start with the following relation

$$(\mathbf{x}^S - \bar{\mathbf{x}}^M) \circ \bar{\mathbf{a}}_\alpha^M = 0 \tag{4.38}$$

The linearization reads

$$\begin{aligned}
(\Delta\mathbf{u}^S - \Delta\bar{\mathbf{u}}^M - \bar{\mathbf{x}}_{,\beta}^M \Delta\bar{\xi}_\beta) \circ \bar{\mathbf{a}}_\alpha^M + (\mathbf{x}^S - \bar{\mathbf{x}}^M) \circ \Delta\bar{\mathbf{a}}_\alpha^M &= 0 \\
(\Delta\mathbf{u}^S - \Delta\bar{\mathbf{u}}^M) \circ \bar{\mathbf{a}}_\alpha^M + (\mathbf{x}^S - \bar{\mathbf{x}}^M) \circ \Delta\bar{\mathbf{a}}_\alpha^M &= (\bar{\mathbf{x}}_{,\beta}^M \Delta\bar{\xi}_\beta) \circ \bar{\mathbf{a}}_\alpha^M \\
(\Delta\mathbf{u}^S - \Delta\bar{\mathbf{u}}^M) \circ \bar{\mathbf{a}}_\alpha^M + (\mathbf{x}^S - \bar{\mathbf{x}}^M) \circ \Delta\bar{\mathbf{a}}_\alpha^M &= (\bar{\mathbf{a}}_\beta^M \Delta\bar{\xi}_\beta) \circ \bar{\mathbf{a}}_\alpha^M \\
(\Delta\mathbf{u}^S - \Delta\bar{\mathbf{u}}^M) \circ \bar{\mathbf{a}}_\alpha^M + g\bar{\mathbf{n}}^M \circ \Delta\bar{\mathbf{a}}_\alpha^M &= \Delta\bar{\xi}_\beta (\bar{\mathbf{a}}_\beta^M \circ \bar{\mathbf{a}}_\alpha^M) \\
(\Delta\mathbf{u}^S - \Delta\bar{\mathbf{u}}^M) \circ \bar{\mathbf{a}}_\alpha^M + g\bar{\mathbf{n}}^M \circ \Delta\bar{\mathbf{u}}_{,\alpha}^M &= \Delta\bar{\xi}_\beta \bar{a}_{\alpha\beta} \\
\left[(\Delta\mathbf{u}^S - \Delta\bar{\mathbf{u}}^M) \circ \bar{\mathbf{a}}_\alpha^M + g\bar{\mathbf{n}}^M \circ \Delta\bar{\mathbf{u}}_{,\alpha}^M\right] \bar{a}^{\alpha\beta} &= \Delta\bar{\xi}_\beta
\end{aligned} \tag{4.39}$$

4.2.2.1.3.4 The term $\delta\bar{\xi}_\alpha$

The variation is now trivial

$$\delta\bar{\xi}_\beta = \bar{a}^{\alpha\beta} \left[(\delta\mathbf{x}^S - \delta\bar{\mathbf{x}}^M) \circ \bar{\mathbf{a}}_\alpha^M + g\bar{\mathbf{n}}^M \circ \delta\bar{\mathbf{x}}_{,\alpha}^M\right] \tag{4.40}$$

4.2.3 Assembly of the linearization

Finally the linearization can be formulated with respect to the increments of the displacements. Due to the quite long derivations it is worth, to summarize all derived equations, before the assembly is done.

Summary

Δg	$= (\Delta\mathbf{u}^S - \Delta\bar{\mathbf{u}}^M) \circ \bar{\mathbf{n}}^M$	equ. (4.23) on p. 28
δg	$= (\delta\mathbf{x}^S - \delta\bar{\mathbf{x}}^M) \circ \bar{\mathbf{n}}^M$	equ. (4.24) on p. 28
$\Delta\delta g$	$= (-\delta\bar{\mathbf{x}}_{,\alpha}^M \Delta\bar{\xi}_\alpha - \Delta\bar{\mathbf{u}}_{,\alpha}^M \delta\bar{\xi}_\alpha) \circ \bar{\mathbf{n}}^M + g\delta\bar{\mathbf{n}}^M \circ \Delta\bar{\mathbf{n}}^M$	equ. (4.31) on p. 29
$\Delta\bar{\mathbf{n}}^M$	$= -(\bar{\mathbf{n}}^M \circ \Delta\bar{\mathbf{u}}_{,\alpha}^M)\bar{a}^{\alpha\beta}\bar{\mathbf{a}}_\beta^M$	equ. (4.36) on p. 29
$\delta\bar{\mathbf{n}}^M$	$= -(\bar{\mathbf{n}}^M \circ \delta\bar{\mathbf{x}}_{,\alpha}^M)\bar{a}^{\alpha\beta}\bar{\mathbf{a}}_\beta^M$	equ. (4.37) on p. 29
$\Delta\bar{\xi}_\beta$	$= \bar{a}^{\alpha\beta}\left[(\Delta\mathbf{u}^S - \Delta\bar{\mathbf{u}}^M) \circ \bar{\mathbf{a}}_\alpha^M + g\bar{\mathbf{n}}^M \circ \Delta\bar{\mathbf{u}}_{,\alpha}^M\right]$	equ. (4.39) on p. 30
$\delta\bar{\xi}_\beta$	$= \bar{a}^{\alpha\beta}\left[(\delta\mathbf{x}^S - \delta\bar{\mathbf{x}}^M) \circ \bar{\mathbf{a}}_\alpha^M + g\bar{\mathbf{n}}^M \circ \delta\bar{\mathbf{x}}_{,\alpha}^M\right]$	equ. (4.40) on p. 30

Now we can assemble all equations we start with $\Delta\delta g$.

$$\begin{aligned}
\Delta\delta g &= (-\delta\bar{\mathbf{x}}^M_{,\alpha}\Delta\bar{\xi}_\alpha - \Delta\bar{\mathbf{u}}^M_{,\alpha}\delta\bar{\xi}_\alpha)\circ\bar{\mathbf{n}}^M + g\delta\bar{\mathbf{n}}^M\circ\Delta\bar{\mathbf{n}}^M && (4.41)\\
&= (-\delta\bar{\mathbf{x}}^M_{,\alpha}\Delta\bar{\xi}_\alpha - \Delta\bar{\mathbf{u}}^M_{,\alpha}\delta\bar{\xi}_\alpha)\circ\bar{\mathbf{n}}^M - g\delta\bar{\mathbf{n}}^M\circ(\bar{\mathbf{n}}^M\circ\Delta\bar{\mathbf{u}}^M_{,\alpha})\bar{a}^{\alpha\beta}\bar{\mathbf{a}}^M_\beta && (4.42)\\
&= (-\delta\bar{\mathbf{x}}^M_{,\alpha}\Delta\bar{\xi}_\alpha - \Delta\bar{\mathbf{u}}^M_{,\alpha}\delta\bar{\xi}_\alpha)\circ\bar{\mathbf{n}}^M\\
&\quad + g(\bar{\mathbf{n}}^M\circ\delta\bar{\mathbf{x}}^M_{,\alpha})\bar{a}^{\alpha\beta}\bar{\mathbf{a}}^M_\beta\circ(\bar{\mathbf{n}}^M\circ\Delta\bar{\mathbf{u}}^M_{,\chi})\bar{a}^{\chi\gamma}\bar{\mathbf{a}}^M_\gamma && (4.43)\\
&= (-\delta\bar{\mathbf{x}}^M_{,\alpha}\Delta\bar{\xi}_\alpha - \Delta\bar{\mathbf{u}}^M_{,\alpha}\delta\bar{\xi}_\alpha)\circ\bar{\mathbf{n}}^M + g\,\bar{a}^{\alpha\chi}(\bar{\mathbf{n}}^M\circ\delta\bar{\mathbf{x}}^M_{,\alpha})(\bar{\mathbf{n}}^M\circ\Delta\bar{\mathbf{u}}^M_{,\chi}) && (4.44)\\
&= (-\delta\bar{\mathbf{x}}^M_{,\alpha}\Delta\bar{\xi}_\alpha - \Delta\bar{\mathbf{u}}^M_{,\beta}\bar{a}^{\alpha\beta}\left[(\delta\mathbf{x}^S - \delta\bar{\mathbf{x}}^M)\circ\bar{\mathbf{a}}^M_\alpha + g\bar{\mathbf{n}}^M\circ\delta\bar{\mathbf{x}}^M_{,\alpha}\right])\circ\bar{\mathbf{n}}^M && (4.45)\\
&\quad + g\,\bar{a}^{\alpha\chi}(\bar{\mathbf{n}}^M\circ\delta\bar{\mathbf{x}}^M_{,\alpha})(\bar{\mathbf{n}}^M\circ\Delta\bar{\mathbf{u}}^M_{,\chi})\\
&= (-\delta\bar{\mathbf{x}}^M_{,\alpha}\Delta\bar{\xi}_\alpha - \Delta\bar{\mathbf{u}}^M_{,\beta}\bar{a}^{\alpha\beta}\left[(\delta\mathbf{x}^S - \delta\bar{\mathbf{x}}^M)\circ\bar{\mathbf{a}}^M_\alpha\right])\circ\bar{\mathbf{n}}^M && (4.46)\\
&= -(\delta\bar{\mathbf{x}}^M_{,\alpha}\Delta\bar{\xi}_\alpha)\circ\bar{\mathbf{n}}^M - (\Delta\bar{\mathbf{u}}^M_{,\beta}\circ\bar{\mathbf{n}}^M)\bar{a}^{\alpha\beta}((\delta\mathbf{x}^S - \delta\bar{\mathbf{x}}^M)\circ\bar{\mathbf{a}}^M_\alpha) && (4.47)\\
&= -\Delta\bar{\xi}_\alpha\delta\bar{\mathbf{x}}^M_{,\alpha}\circ\bar{\mathbf{n}}^M - \bar{a}^{\alpha\beta}(\delta\mathbf{x}^S - \delta\bar{\mathbf{x}}^M)\circ(\bar{\mathbf{a}}^M_\alpha\otimes\bar{\mathbf{n}}^M)\Delta\bar{\mathbf{u}}^M_{,\beta} && (4.48)\\
&= -\Delta\bar{\xi}_\alpha\delta\bar{\mathbf{x}}^M_{,\alpha}\circ\bar{\mathbf{n}}^M - \bar{a}^{\alpha\beta}(\delta\mathbf{x}^S - \delta\bar{\mathbf{x}}^M)\circ(\bar{\mathbf{a}}^M_\alpha\otimes\bar{\mathbf{n}}^M)\Delta\bar{\mathbf{a}}^M_\beta && (4.49)
\end{aligned}$$

We can now start to assemble the last equation. This is the linearization of the first variation of equation (4.20) on page 27. Combining equation (4.49) and equation (4.39) on the previous page yields:

$$\begin{aligned}
g\Delta\delta g = g\Big[&-\bar{a}^{\alpha\beta}\left[(\Delta\mathbf{u}^S - \Delta\bar{\mathbf{u}}^M)\circ\bar{\mathbf{a}}^M_\alpha + g\bar{\mathbf{n}}^M\circ\Delta\bar{\mathbf{u}}^M_{,\alpha}\right]\delta\bar{\mathbf{x}}^M_{,\beta}\circ\bar{\mathbf{n}}^M\\
&-\bar{a}^{\alpha\beta}(\delta\mathbf{x}^S - \delta\bar{\mathbf{x}}^M)\circ(\bar{\mathbf{a}}^M_\alpha\otimes\bar{\mathbf{n}}^M)\Delta\bar{\mathbf{a}}^M_\beta\Big]
\end{aligned} \tag{4.50}$$

Lets separate all terms belonging to g and g^2, respectively.

$$\begin{aligned}
&g\Delta\delta g =\\
&g\left[-\bar{a}^{\alpha\beta}\left[(\Delta\mathbf{u}^S - \Delta\bar{\mathbf{u}}^M)\circ\bar{\mathbf{a}}^M_\alpha\right]\delta\bar{\mathbf{x}}^M_{,\beta}\circ\bar{\mathbf{n}}^M - \bar{a}^{\alpha\beta}(\delta\mathbf{x}^S - \delta\bar{\mathbf{x}}^M)\circ(\bar{\mathbf{a}}^M_\alpha\otimes\bar{\mathbf{n}}^M)\Delta\bar{\mathbf{a}}^M_\beta\right]\\
&- g^2\bar{a}^{\alpha\beta}\left[\bar{\mathbf{n}}^M\circ\Delta\bar{\mathbf{u}}^M_{,\alpha}\right]\delta\bar{\mathbf{x}}^M_{,\beta}\circ\bar{\mathbf{n}}^M
\end{aligned} \tag{4.51}$$

It is possible to reformulate equation (4.51) by using $(\mathbf{l}\otimes\mathbf{m})\mathbf{o} = (\mathbf{o}\circ\mathbf{m})\mathbf{l}$ (see also [7, page 21])

$$\begin{aligned}
&g\Delta\delta g =\\
&-g\left[\delta\bar{\mathbf{x}}^M_{,\alpha}(\bar{\mathbf{n}}^M\otimes\bar{a}^{\alpha\beta}\bar{\mathbf{a}}^M_\beta)\circ(\Delta\mathbf{u}^S - \Delta\bar{\mathbf{u}}^M) + (\delta\mathbf{x}^S - \delta\bar{\mathbf{x}}^M)\circ(\bar{a}^{\alpha\beta}\bar{\mathbf{a}}^M_\alpha\otimes\bar{\mathbf{n}}^M)\Delta\bar{\mathbf{a}}^M_\beta\right]\\
&- g^2\left[\bar{a}^{\alpha\beta}\delta\bar{\mathbf{x}}^M_{,\alpha}(\bar{\mathbf{n}}^M\otimes\bar{\mathbf{n}}^M)\circ\Delta\bar{\mathbf{a}}^M_\beta\right]
\end{aligned} \tag{4.52}$$

For one contact element we now have for the linearized variation

$$\begin{aligned}\Delta\delta\Pi^{C\,e}_{Penalty} = cA_n \Big\{ &(\delta\mathbf{x}^S - \delta\bar{\mathbf{x}}^M) \circ (\bar{\mathbf{n}}^M \otimes \bar{\mathbf{n}}^M)(\Delta\mathbf{u}^S - \Delta\bar{\mathbf{u}}^M) \\ &- g \left[\delta\bar{\mathbf{x}}^M_{,\alpha} (\bar{\mathbf{n}}^M \otimes \bar{a}^{\alpha\beta}\bar{\mathbf{a}}^M_\beta) \circ (\Delta\mathbf{u}^S - \Delta\bar{\mathbf{u}}^M) \right. \\ &\left. + (\delta\mathbf{x}^S - \delta\bar{\mathbf{x}}^M) \circ (\bar{a}^{\alpha\beta}\bar{\mathbf{a}}^M_\alpha \otimes \bar{\mathbf{n}}^M)\Delta\bar{\mathbf{a}}^M_\beta \right] \\ &- g^2 \left[\bar{a}^{\alpha\beta}\delta\bar{\mathbf{x}}^M_{,\alpha} (\bar{\mathbf{n}}^M \otimes \bar{\mathbf{n}}^M) \circ \Delta\bar{\mathbf{a}}^M_\beta \right] \Big\} \end{aligned} \tag{4.53}$$

By introducing the contravariant base vectors $\bar{a}^{\alpha\beta}\bar{\mathbf{a}}^M_\alpha = \bar{\mathbf{a}}^{M\beta}$ and recalling equation (4.35) on page 29 the final form of equation (4.20) on page 27 is obtained.

$$\begin{aligned}\Delta\delta\Pi^{C\,e}_{Penalty} = cA_n \Big\{ &(\delta\mathbf{x}^S - \delta\bar{\mathbf{x}}^M) \circ (\bar{\mathbf{n}}^M \otimes \bar{\mathbf{n}}^M)(\Delta\mathbf{u}^S - \Delta\bar{\mathbf{u}}^M) \\ &- g \left[\delta\bar{\mathbf{x}}^M_{,\alpha} (\bar{\mathbf{n}}^M \otimes \bar{\mathbf{a}}^{M\alpha}) \circ (\Delta\mathbf{u}^S - \Delta\bar{\mathbf{u}}^M) \right. \\ &\left. + (\delta\mathbf{x}^S - \delta\bar{\mathbf{x}}^M) \circ (\bar{\mathbf{a}}^{M\beta} \otimes \bar{\mathbf{n}}^M)\Delta\bar{\mathbf{u}}^M_{,\beta} \right] \\ &- g^2 \left[\bar{a}^{\alpha\beta}\delta\bar{\mathbf{x}}^M_{,\alpha} (\bar{\mathbf{n}}^M \otimes \bar{\mathbf{n}}^M) \circ \Delta\bar{\mathbf{u}}^M_{,\beta} \right] \Big\} \end{aligned} \tag{4.54}$$

4.2.4 Residual vector and tangent stiffness matrix

First let us introduce some vectors:

$$\mathbf{m}_f = \begin{bmatrix} \bar{\mathbf{n}}^M \\ -N_1\bar{\mathbf{n}}^M \\ -N_2\bar{\mathbf{n}}^M \\ -N_3\bar{\mathbf{n}}^M \end{bmatrix}, \mathbf{m}_\alpha = \begin{bmatrix} \mathbf{0} \\ -N_{1,\alpha}\bar{\mathbf{n}}^M \\ -N_{2,\alpha}\bar{\mathbf{n}}^M \\ -N_{3,\alpha}\bar{\mathbf{n}}^M \end{bmatrix}, \mathbf{t}_\beta = \begin{bmatrix} \bar{\mathbf{a}}^{M\beta} \\ -N_1\bar{\mathbf{a}}^{M\beta} \\ -N_2\bar{\mathbf{a}}^{M\beta} \\ -N_3\bar{\mathbf{a}}^{M\beta} \end{bmatrix} \tag{4.55}$$

For the unknowns we introduce:

$$\delta\mathbf{x}_f = \begin{bmatrix} \delta\mathbf{x}^S \\ \delta\mathbf{x}^M_1 \\ \delta\mathbf{x}^M_2 \\ \delta\mathbf{x}^M_3 \end{bmatrix}, \Delta\mathbf{u}_f = \begin{bmatrix} \Delta\mathbf{u}^S \\ \Delta\mathbf{u}^M_1 \\ \Delta\mathbf{u}^M_2 \\ \Delta\mathbf{u}^M_3 \end{bmatrix} \tag{4.56}$$

With these vectors we can rewrite equation (4.54) to

$$\Delta\delta\Pi^{C\,e}_{Penalty} = \delta\mathbf{x}_f^T \mathbf{K}^C_T \Delta\mathbf{u}_f$$

and equation (4.19) on page 27 reads:

$$\delta\Pi^{C\,e}_{Penalty} = \delta\mathbf{x}_f^T \mathbf{r}^C$$

So residual vector for the 4-node contact element is defined by:

$$\mathbf{r}^C = c\ A_n\ g\ \mathbf{m}_f \tag{4.57}$$

and the tangent stiffness matrix is

$$\mathbf{K}_T^C = \overbrace{c\, A_n \left\{ (\mathbf{m}_1 \otimes \mathbf{m}_1) }^{\text{linear part}} - g \left[(\mathbf{m}_1 \otimes \mathbf{t}_1) + (\mathbf{m}_2 \otimes \mathbf{t}_2) + (\mathbf{t}_1 \otimes \mathbf{m}_1) + (\mathbf{t}_2 \otimes \mathbf{m}_2) + g \left(a^{11}(\mathbf{m}_1 \otimes \mathbf{m}_1) + a^{12}(\mathbf{m}_1 \otimes \mathbf{m}_2 + \mathbf{m}_2 \otimes \mathbf{m}_1) + a^{22}(\mathbf{m}_2 \otimes \mathbf{m}_2) \right) \right] \right\} \quad (4.58)$$

4.3 Summary

The penalty 4-node contact element can be integrated in a FE-framework almost exactly as a nonlinear structural element. Although the change of contact status during the solution iterations yields to a severe discontinuity in the solution space. Therefore such changes are also known as SDIs (Severe Discontinuity Iterations). Because of the possibility of contact status change, contact elements need to be treated special. For further information please refer to chapter 5.

A nice property of equation (4.58) is that it is symmetric in the displacement increments $\Delta \mathbf{u}_f$. That allows the use of a symmetric solver for each iteration. Note that most of the derivations in this chapter can be proven mathematically with the help of [7, chapter 1].

As the gap function is not differentiable the quadratic convergence can only be reached when the contact status is not changing anymore (no SDIs). Because just the linear term in equation (4.58) does not depend on the gap g, this term can be used in the first stages of the Newton-Raphson iterations, where SDIs occur to approximate the tangent stiffness matrix. Because of the dependence of the other terms on g they can get quite large within the first iterations and lead to divergence. Another reason for an approximation of the tangent during the first iterations is that the tangent matrix (equation (4.58)) is in general not positive-semidefinite. It is indefinite, hence it has negative eigenvalues. This is mainly caused by the second term. For a high penalty stiffness in combination with large overclosures in the first Newton-Raphson iterations, this can take preeminence and lead to divergence.

Chapter 5

Implementation in Carat++

This chapter deals with implementation details. The nonlinear object-oriented finite element software Carat++ will be used as a basis for the implementation. Due to the class structure (see figure 5.1) the contact can be implemented without major changes in the code structure. The contact element can in general be

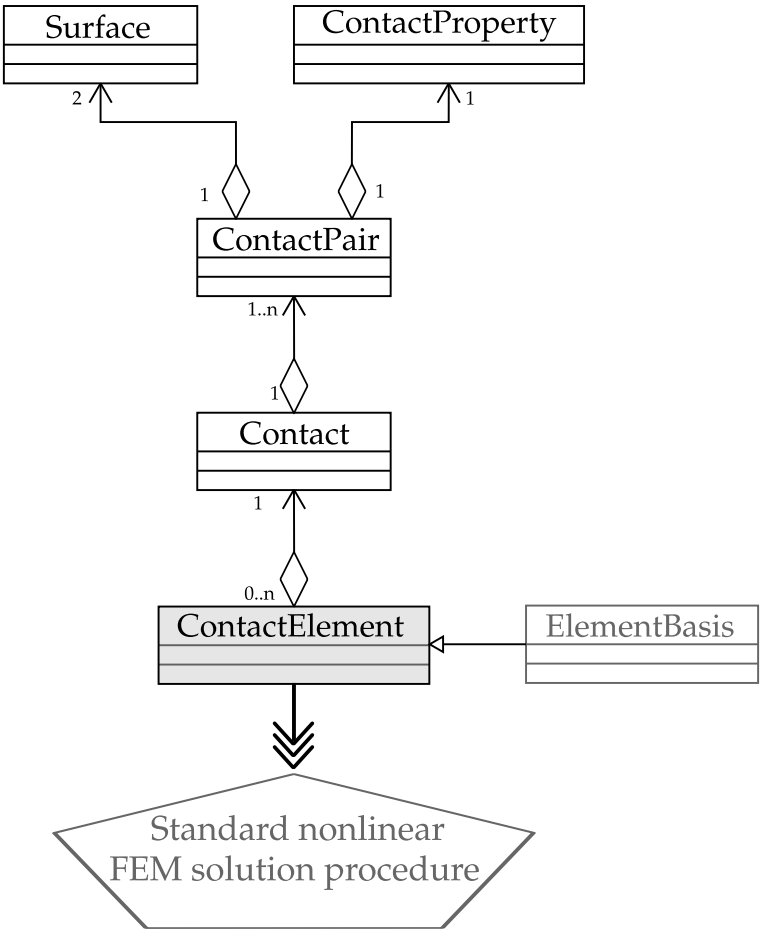


Figure 5.1: UML like diagram of object-oriented mechanical contact implementation

treated as a standard element (truss, shell or continuum). On top of the standard properties the contact element has the ability to change its status from active to

deactive or the other way around. This is one of the major reasons, why contact problems are difficult to solve.
1.1

5.1 Surface class

The surface class is used in order to be able to define the potential contact partners. There are two types of surfaces. One is node-based, which means that the surface is a cloud of nodes. The other one is face-based, that means the information of the underlying element topology can be reconstructed.

5.1.1 User input block for the surface class

The node-based surface is used for the slave surface. The user has to provide a unique surface ID and a list of nodes which build up the slave surface.

Inputblock 5.1: Node-based surface

```
SURFACE 1 : NODE
1, 2, 3, 4, 5,
6, 7, 8, 9
```

The face-based surface also needs a unique identifier. Here the data is not only a node cloud. The user must specify for each element a face identifier.

Inputblock 5.2: Face-based surface

```
SURFACE 2 : ELEMENT
1, S1
3, S2
```

Let us assume that element 1 and 3 are triangular elements, in this case element 1 would define the contact surface by a positive normal and element 3 by a negative normal. This concept can easily be expanded to continuum elements.

5.1.2 Implementation details for the surface class

The surface identifiers for the triangular elements are such that a mathematical positive node numbering defines the normal . This means that if the user would like to have the red face of the element shown in figure 5.2 on the next page as part of the contact surface he must provide S1 for this element. If the user would like the contact surface to be build up by the bottom face of the element (not visible in figure 5.2 on the following page) he or she has to provide S2 for this element.

5.1.3 Applications and limitations of the implementation

At the moment the surface concept can only be used with contact. The face identifiers do only work with triangular elements. Surfaces can only be used for contact master surfaces.

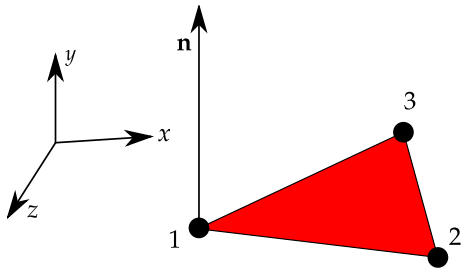


Figure 5.2: Surface identifiers for a triangular element

5.1.4 Possible enhancements

The surface concept could be generalized to surface coupled problems(like fluid-structure interaction). It can also be used for surface loads (like pressure or snow). The surface identifiers should become a standard data member of each element.

5.2 Contact property class

This class defines the physical properties of the contact and it defines the constraint enforcement method. Depending on the enforcement method the corresponding parameters are also defined within this class.

5.2.1 User input block for the contact property class

The input block for the contact property block is called SURFACE-INTERACTION, indicating that it could be used also for other surface-coupled problems. Within this input block the user must also specify a unique identifier, because it is possible to have more SURFACE-INTERACTION blocks within one analysis.

Inputblock 5.3: Surface interaction block defines contact properties

```
SURFACE-INTERACTION 1
PHYSICAL-NORMAL-BEHAVIOR=HARD
PHYSICAL-TANGENTIAL-BEHAVIOR=NOFRICTION
CONSTRAINT-ENFORCEMENT-NORMAL=PENALTY_CONST
PENALTY_PARAMETER=1e5
```

5.2.2 Implementation details for the contact property class

So far only four parameters are implemented. The first one is the PHYSICAL-NORMAL-BEHAVIOR. At the moment it can just deal with one option (HARD). This parameter does not indicate which enforcement method is used, it defines the physical normal behavior. There are situations, where the contact constraint is not always a hard enforced constraint. In fact in nature the contact phenomena is a "soft" process. This is because of the microstructure of the contact surfaces (they are never flat). If the user would like to account this phenomena, it must be possible to define a force-displacement curve until the bodies are in full contact

and hence are supported by the underlying stiffness of the deformable bodies. A further application could be, if surface coating should be taken into account. As a counterpart for the normal direction a second parameter must be specified, which defines the tangential behavior (PHYSICAL-TANGENTIAL-BEHAVIOR). So far only frictionless contact is possible. Last but not least parameters for the chosen enforcement method must be specified. For this implementation only the penalty method is possible.

5.2.3 Applications and limitations of the implementation

Some of the limitations have already been discussed in the previous subsection. Therefore we will only discuss the limitations in combination with the penalty method here. The penalty method needs a penalty stiffness, which can in general be a function of the gap. At the moment only a constant penalty stiffness can be provided. Equation (4.58) on page 33 needs also the associated slave surface area for each contact element A_n. This is assumed to be one at the current state of the implementation.

5.2.4 Possible enhancements

The list for possible enhancements associated with a SURFACE-INTERACTION object is very long and not everything can be discussed. The least implementation efforts are to implement the dependency on the slave surface area and different dependencies of the penalty stiffness on the gap.

The contact could be expanded by tangential contributions to the functional, which is e.g. friction. There are different frictional models, which could be implemented. The normal enforcement methods could be enriched by Lagrange multiplier methods and augmented Lagrange methods.

5.3 Contact pair class

This class needs two surface objects (type NODE and type ELEMENT) and one contact property object (see also figure 5.1 on page 34). This class performs the contact search.

5.3.1 User input block for the contact pair class

The user must specify a SLAVE-SURFACE, which is of type NODE and a MASTER-SURFACE, which is of type ELEMENT. Furthermore a link to the contact property class has to be provided. A crucial option is the CONTACT-TRACKING approach. SMALL-SLIDING means that the contact pairings are kept constant within one load increment. FINITE-SLIDING on the other hand means, that the pairings are redetermined at each iteration of the nonlinear solution procedure.

The CONTACT-DISCRETIZATION methods could be expanded to SURFACE-TO-SURFACE methods (e.g. mortar methods). At the moment only NODE-TO-SURFACE is implemented.

Inputblock 5.4: Contact pair with its properties

```
CONTACT-PAIR 1
SLAVE-SURFACE = SURFACE 1
MASTER-SURFACE = SURFACE 2
CONTACT-PROPERTIES = SURFACE-INTERACTION 1
CONTACT-TRACKING = SMALL-SLIDING !FINITE-SLIDING
CONTACT-DISCRETIZATION = NODE-TO-SURFACE
```

5.3.2 Implementation details for the contact pair class

The contact searching is the most computational effort in this class. It consists of a nested loop (one runs over master faces the other over the slave nodes), which projects each slave node on each master face. In the following it is checked, if the projected slave node is inside the master face. If so, a contact element will be generated for this slave node master face pairing (see algorithm 3). At the moment only SMALL-SLIDING is implemented. The big advantage of SMALL-SLIDING is that the matrix structure is preserved during each load increment. In order to be able to take SDIs into account the contact elements, which have a open contact status get a zero tangent stiffness and a zero residual force vector.

Algorithm 3 The contact search algorithm

1: $TOL \leftarrow 0.000000001$
2: **for** $i \leftarrow 1, num\_slave\_nodes$ **do**
3: **for** $j \leftarrow 1, num\_master\_faces$ **do**
4: $slave\_coord \leftarrow$ GETSLAVECOORD(i)
5: ▷ Get slave node coordinates for slave node i
6: $master\_coord \leftarrow$ GETMASTERCOORD(j)
7: ▷ Get all master face node coordinates for master face j
8: $\tilde{\xi_1}, \tilde{\xi_2} \leftarrow$ COMPUTEPROJECTION($slave\_coord$, $master\_coord$)
9: $N_1, N_2, N_3 \leftarrow$ EVALUATESHAPEFUNCTIONS($\tilde{\xi_1}, \tilde{\xi_2}$)
10: **if** $(N_1 \geq -TOL$ **and** $N_1 \leq 1 + TOL)$ **and** $(N_2 \geq -TOL$ **and** $N_2 \leq 1 + TOL)$ **and** $(N_3 \geq -TOL$ **and** $N_3 \leq 1 + TOL)$ **then**
11: **return** CONTACTPAIRFOUND ▷ A valid contact pair is found
12: **end if**
13: **end for**
14: **end for**

5.3.3 Applications and limitations of the implementation

The FINITE-SLIDING approach does not work at the moment. FINITE-SLIDING demands a new tangent matrix structure in each iteration and a redetermination

of contact pairs in each iteration. The implemented contact searching algorithm is a rather simple approach and could get in troubles, if the difference in the normal between aligned master faces is too large.

5.3.4 Possible enhancements

As a remedy of the previous subsections a more sophisticated contact searching algorithm would be one possible enhancement. Also the possibility to change the matrix structure from iteration to iteration is a good enhancement in order to be able to model contact problems with large tangential sliding.

Adding different contact discretization schemes (like mortar based methods) is not a straight forward task and would take considerably more effort.

5.4 Contact class

The contact class is the master class for contact. It has access to the hole structure of the contact problem. It can be linked to several contact pair objects.

5.4.1 User input block for the contact class

The contact class expects a link (or several links) to contact pair objects.

Inputblock 5.5: Contact pair

```
CONTACT 1
CONTACT-PAIRS = CONTACT-PAIR 1
```

In order to activate contact the user must activate the contact object in the analysis input block.

Inputblock 5.6: Contact activation flag in the analysis block

```
PC-ANALYSIS 1: STA_GEO_NONLIN
  PATHCONTROL = FORCE ! or DISPLACEMENT or ARCLENGTH
  SOLVER = PC-SOLVER 2
  OUTPUT = PC-OUT 1
  COMPCASE = LD-COM 1
  DOMAIN = EL-DOMAIN 1
  NUM_STEP = 1
  MAX_ITER_EQUILIBRIUM = 20
  EQUILIBRIUM_ACCURACY = 1e-08
  CURVE = LD-CURVE 1
  TRACED_NODE = 39
  TRACED_NODAL_DOF = DISP_Y
  CONTACT = CONTACT 1
```

5.4.2 Implementation details for the contact class

The contact object delivers a vector of active contact elements to the nonlinear static analysis.

5.4.3 Applications and limitations of the implementation

The present implementation can only handle static nonlinear problems, but it could be expanded to dynamic applications. For implicit dynamic problems the contact front has to be tracked within the time increments in order to prevent excessively large penetrations.

5.5 Contact element class

The contact element class delivers the tangent stiffness matrix and residual force vector (see equation (4.58) on page 33 and equation (4.57) on page 32) for each contact element. See chapter 4 in order to get insight into the implementation.

5.6 Summary

The implemented contact algorithm can be characterized by nonlinear kinematics, by the use of the penalty method for constraint enforcement and by the contact discretization, which is node-to-surface. The object structure should easily allow to extend the implemented contact algorithm in various directions. The provided object structure could also be extended to the superior class of surface-coupled problems, where mechanical contact is one special case.

This implementation is a start point for contact mechanics in CARAT++. It should be expanded by several enhancements in order to be able to model complicated physical phenomena. For more information please see also appendix C.

Chapter 6

Results and conclusion

In the following some classical and non-classical contact examples are discussed. At the end of the chapter a conclusion for the implemented contact algorithm is provided.

6.1 Shell versus truss

The first example is the simplest. A truss penetrates in the initial configuration a triangular shell element through its center of gravity. In this scenario only one contact element will be generated. The Carat++ input file and the contact element tangent stiffness matrix for the first iteration are stated in appendix C on page 67 for this problem. If contact is activated the shell element starts to bend and the truss is subjected to a compressive load. The configuration in equilibrium is sketched in figure 6.1. The iteration history for this simple test case is stated in

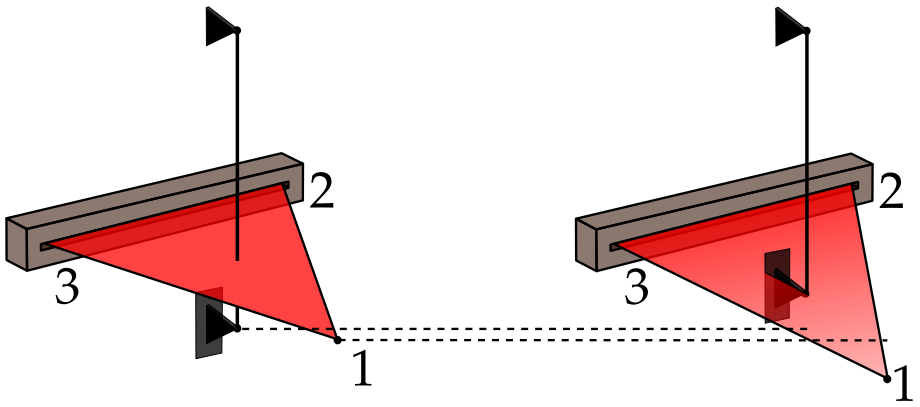


Figure 6.1: The setting for the "shell versus truss" example (undeformed and deformed)

table 6.1 on the following page. The convergence rate is not fully quadratic because the shell element does not have a plane face. If the shell element is replaced by a face of a linear tetrahedral element, the full quadratic convergence rate is obtained. By looking at the support forces of node two and three the symmetry is preserved within round-off errors.

Table 6.1: Iteration history for the "shell versus truss" example

iteration	$\|\mathbf{r}\|_2$	$\|\Delta\mathbf{u}\|_2$	gap
	N	m	m
0			−0.010000
1	0.01054093	0.02628460	−0.000636
2	0.00015114	0.00129987	−0.000637
3	0.00000038	0.00000608	−0.000636
4	0.00000000	0.00000002	−0.000636
5	0.00000000	0.00000000	−0.000636

6.2 Contact patch test

The contact patch test is a method to check, if the contact formulation is able to transfer a constant stress field. If two bodies, which are in contact are subjected to a spatially constant stress field the contact formulation should be able to exactly transmit this stress field from one body to another. If the formulation can do so for arbitrary non-conforming contact interfaces, the contact formulation passes the patch test [16, 34, 36].

The setting of the contact patch is illustrated in figure 6.2. The green upper block is referred to as top block in the following, the blue one as bottom block.

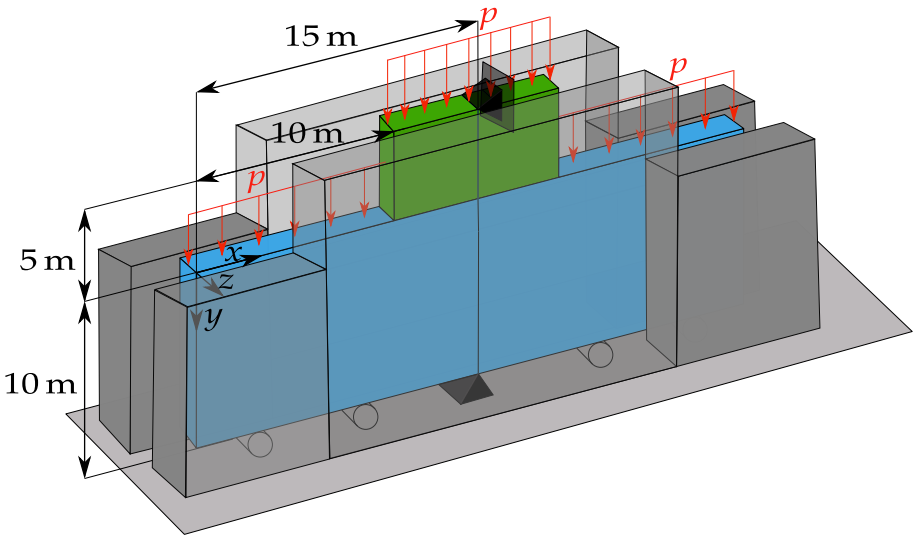


Figure 6.2: The setting for the contact patch test

6.2.1 Finite element solution to the problem

In order to be able to use the triangle based contact algorithm, the contact faces of the brick elements at the master side were equipped with an additional layer of

thin shell elements (see figure 6.3 for top master triangulation and figure 6.4 for the bottom triangulation).

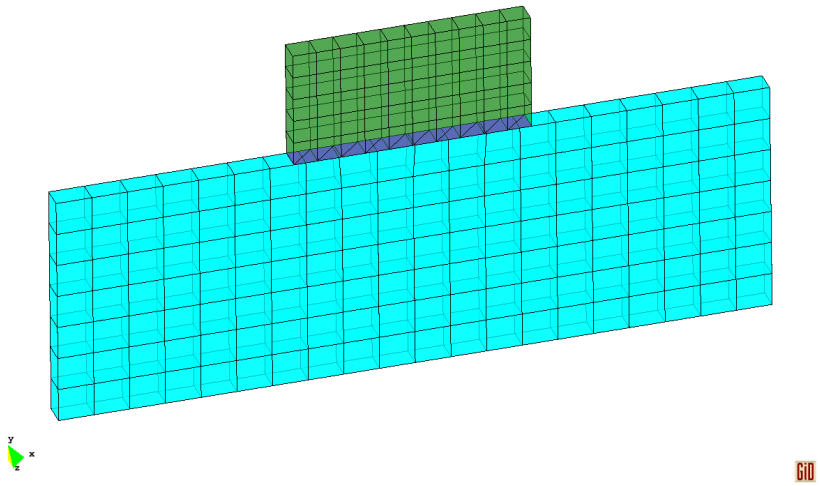


Figure 6.3: The triangulated master surface of the top block

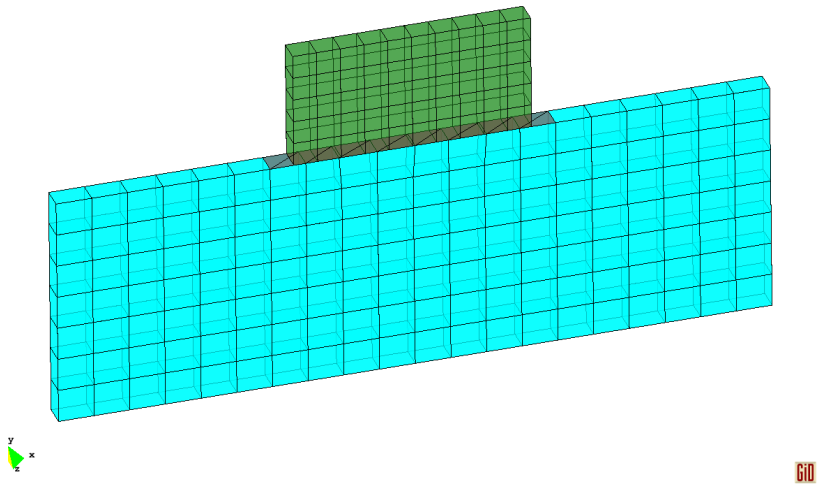


Figure 6.4: The triangulated master surface of the bottom block

The contact algorithm is tested with three different settings (see table 6.2 on the following page). At first the easiest approach is taken. The lower surface (bottom block) is used for the master and the upper one for the slave surface. In general one should use the coarsest mesh as master surfaces. This avoids excessive penetrations. For the first setting the stress in y-direction σ_y is plotted in figure 6.5 on the next page the undeformed configuration is also indicated. The closeup of the contact zone (figure 6.6 on the following page) shows clearly a zigzagging in the displacements.

Table 6.2: Chosen values for different contact patch test settings

Setting ID	E	p	c	ν	Master	Slave
	N/m^2	N/m^2	N/m^2			
1	100	2	$1.0 \cdot 10^5$	0	bottom	top
2	100	2	$1.0 \cdot 10^5$	0	bottom/top	top/bottom
3	100	2	$1.0 \cdot 10^{10}$	0	bottom/top	top/bottom

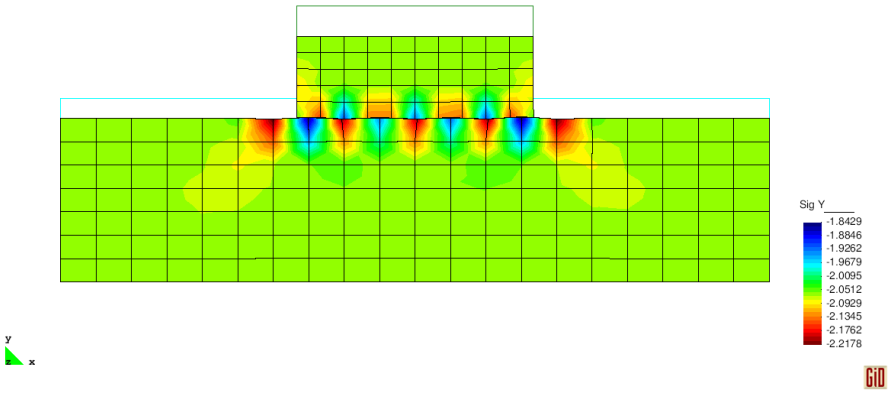


Figure 6.5: σ_y for the patch test of setting 1 (deformation scale factor is 15)

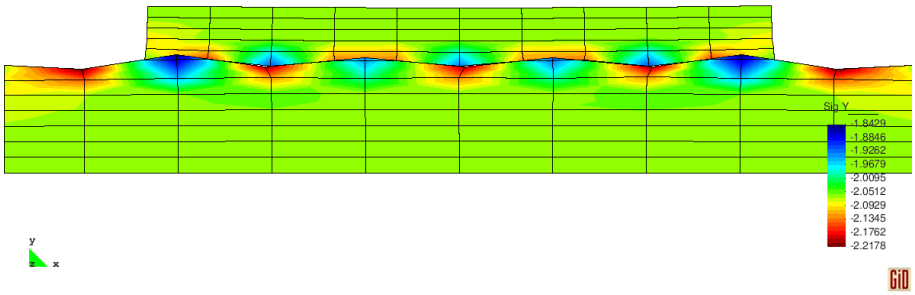


Figure 6.6: σ_y for the patch test of setting 1 (deformation scale factor is 45)

For the second setting a double contact is defined and the penalty stiffness is kept constant in comparison with setting 1. Double contact means, that the lower surface is used as master surface and the upper one as slave surface and vice versa for the second contact pair. With this approach there is no zigzagging anymore in the contact zone (see figure 6.7 on the next page) but the patch test is still not fulfilled. For the last setting the penalty stiffness is adjusted in comparison

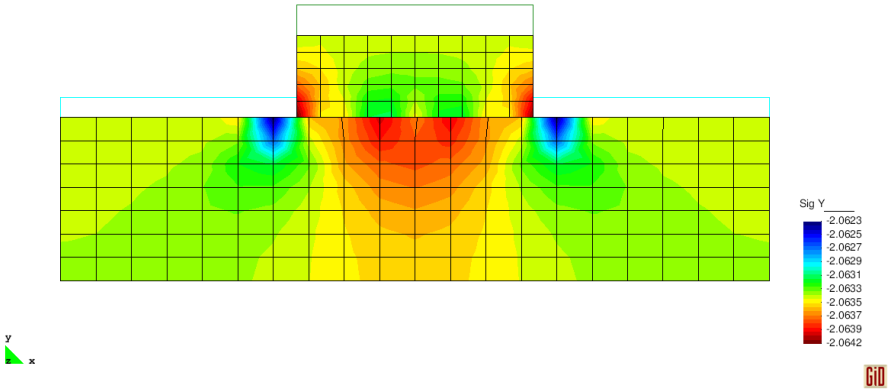


Figure 6.7: σ_y for the patch test of setting 2 (deformation scale factor is 15)

to setting 2. By using the new value for the penalty stiffness (see table 6.2 on the preceding page) and a double contact the patch test is passed to machine precision (see figure 6.8).

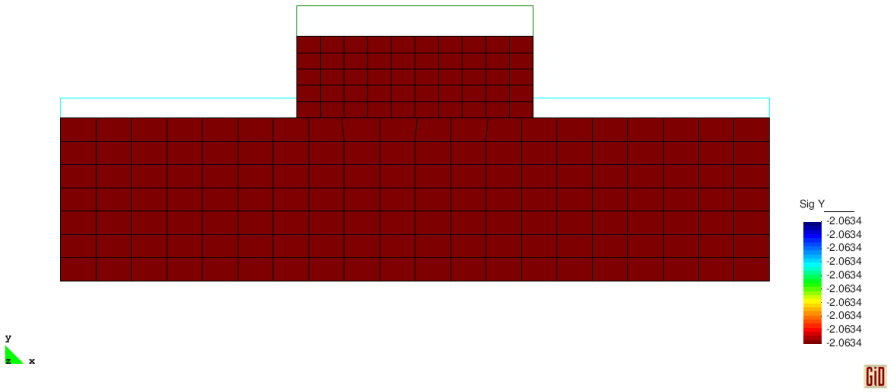


Figure 6.8: σ_y stress for the patch test of setting 3 (deformation scale factor is 15)

6.2.2 Discussion of results

Setting 1 clearly fails the patch test. This is not directly related to the penalty method, it is mainly a problem coming from the node-to-surface discretization. The structure does not need much energy to go from a flat configuration to the zigzagging configuration, even if there would no slave node penetration be allowed (e.g. by Lagrange multipliers). Small disturbances during the iteration process are sufficient to cause slight zigzagging which is not reduced by the contact pressure. The main problem is that only slave node versus master face penetration is checked by the node-to-surface discretization. The node-to-surface discretization does not care about master nodes which are penetrating the slave surface (this is illustrated in figure 6.9 on the following page). The deformation

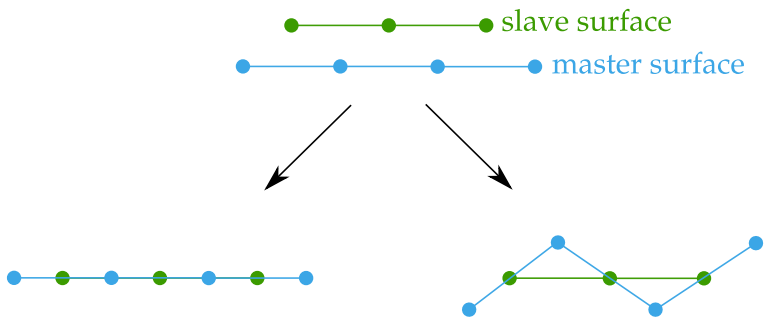


Figure 6.9: Allowed configurations for node-to-surface discretization

pattern is recognized easily by looking at figure 6.10. If double contact is defined the severe zigzagging pattern is suppressed. The slight noise in the stress field is induced by slightly different slave node penetrations coming from the penalty method. If the penalty stiffness is adopted to a fairly large value, the contact patch test is finally passed.

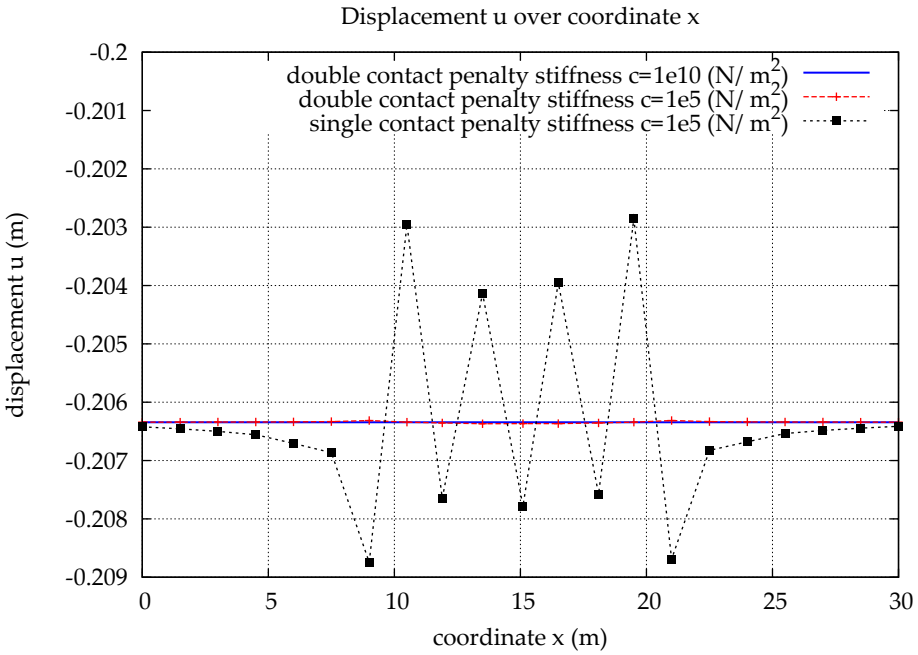


Figure 6.10: Displacement u of the contact interface for different settings

6.3 Hertzian contact problem

Heinrich Hertz derived [11] an analytical solution for bilateral and unilateral contact problems. He made the following assumptions:

- linear elastic, homogeneous and isotropic material
- plain contact zone which is small compared to the dimensions of the contact bodies

- frictionless contact → no shear stress in the contact zone
- contact bodies are regarded as half spaces
- no adhesive forces (e.g. van der Waals forces)

6.3.1 Cylinder versus cylinder

For the problem, where two infinitely long cylinders come in contact, the Hertz solution for the maximal contact pressure $\sigma_{y_{max}}$ reads:

$$\sigma_{y_{max}} = \sqrt{\frac{fE}{2\pi r l_z (1-\nu^2)}} \tag{6.1}$$

where the generalized Young's modulus E, generalized radius r and generalized Poisson's ratio ν are defined by:

$$\begin{aligned} E &= 2\frac{E_1 E_2}{E_1 + E_2} \\ r &= \frac{r_1 r_2}{r_1 + r_2} \\ 1-\nu^2 &= \frac{E}{2}\left(\frac{1-\nu_1^2}{E_1} + \frac{1-\nu_2^2}{E_2}\right) \end{aligned}$$

Furthermore Hertz derived a formula for the width of the contact zone (which has rectangular shape). The width is denoted by s and can be found by:

$$s = \sqrt{\frac{8fr(1-\nu^2)}{\pi E l_z}} \tag{6.2}$$

It is worth to be mentioned, that the contact pressure distribution has circular shape over x. Figure 6.11 on the following page shows the setting. For the reference examples the values from table 6.3 were selected. With these values the

Table 6.3: Chosen values for the example problem

E_1	E_2	r_1	r_2	ν_1	ν_2	l_z	f
N/m²	N/m²	m	m			m	N/m
1	1	5	5	0	0	0.01	$4.0 \cdot 10^{-6}$

derived quantities can be computed (see table 6.4 on the following page).

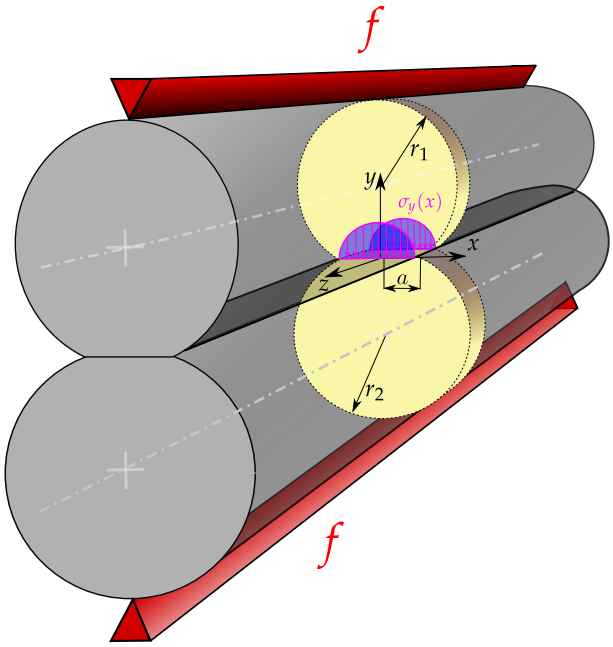


Figure 6.11: Setting for the Hertz contact case "elastic cylinder versus elastic cylinder"

Table 6.4: Analytical solution to the example problem

$\sigma_{y_{max}}$	s
N/m^2	m
0.00504	0.05046

6.3.1.1 Finite element solution to the problem

The finite element model is built up by linear brick elements (standard Galerkin elements). The mesh has only one element in z-direction. The fact that $\nu = 0$ assures, that one can achieve a plain strain and plain stress state at the same time.

This contact problem is symmetric, therefore only half of the geometry is modeled. The mesh in the potential contact zone is shown in figure 6.12. As the imple-

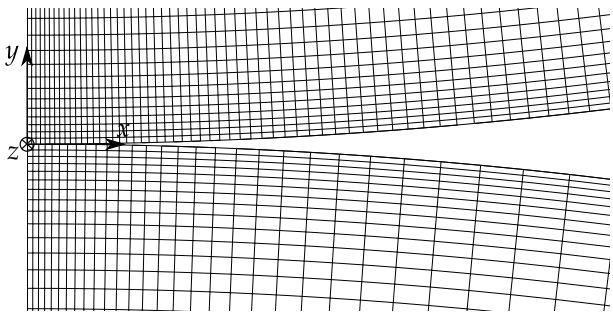
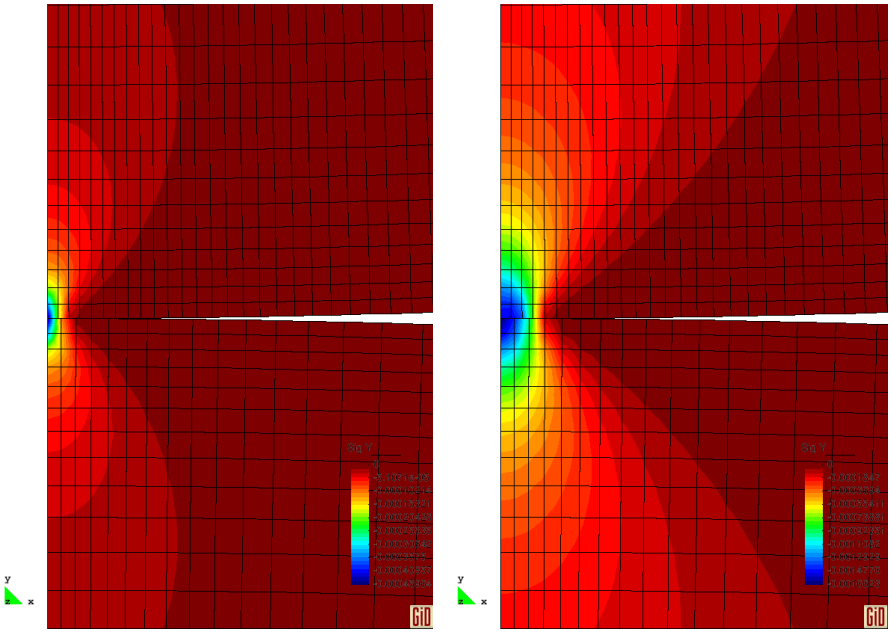


Figure 6.12: Mesh for the Hertzian contact problem cylinder vs. cylinder

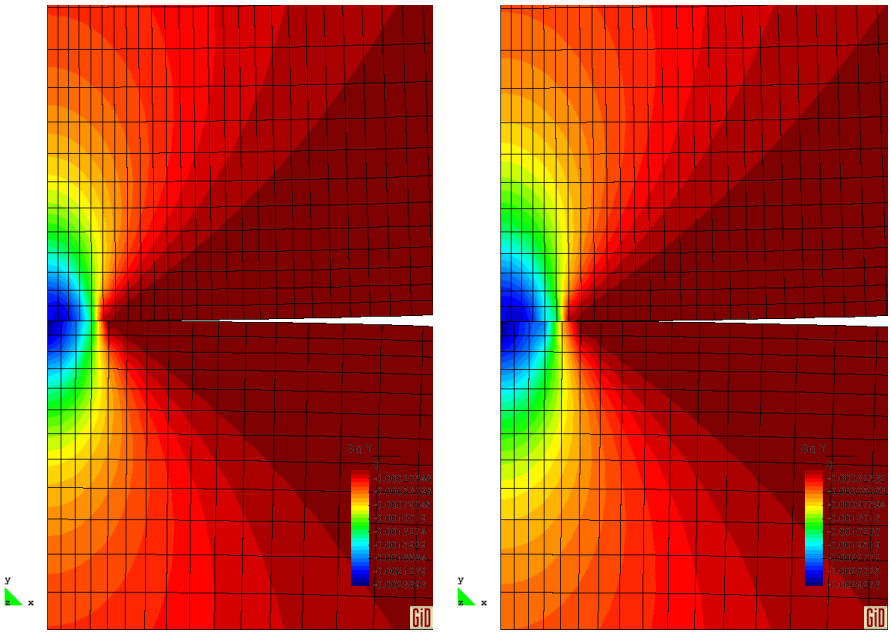
mented contact algorithm does only work with a triangulated master surface, the

same trick as for the contact patch test is applied, in oder to be able to simulate this setting. This procedure shows good convergence in a static simulation.

Figures 6.13a to 6.13k show the evolution of the stress in y-direction σ_y over the pseudo time. The load is ramped up by 1 % per second pseudo time.

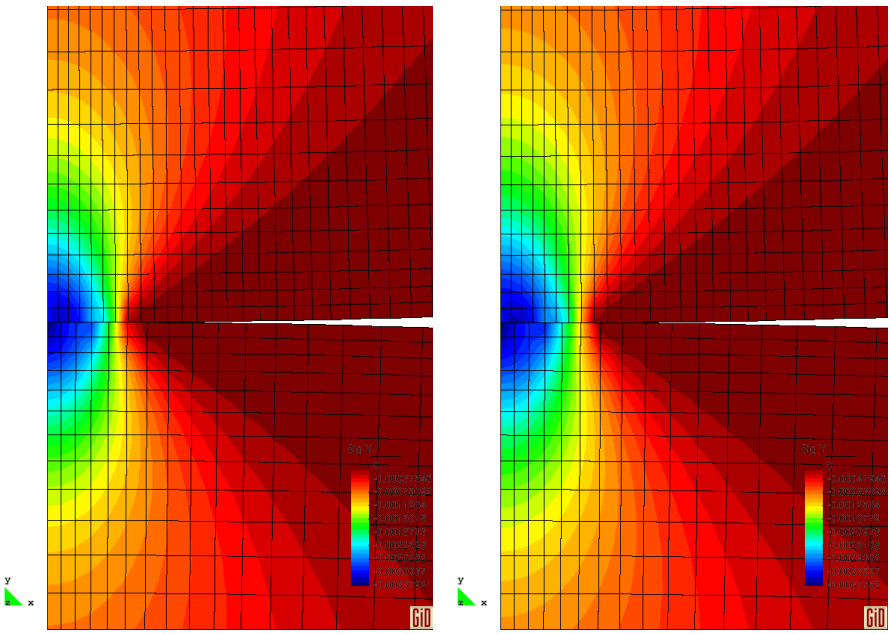


(a) 1 s **(b)** 10 s

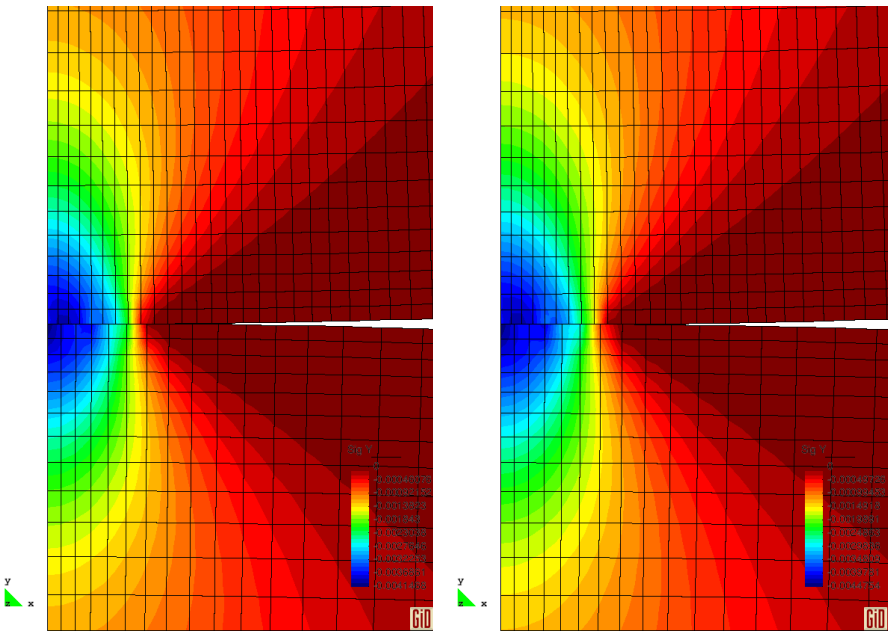


(c) 20 s **(d)** 30 s

Figure 6.13: Evolution of σ_y and contact status over pseudo time

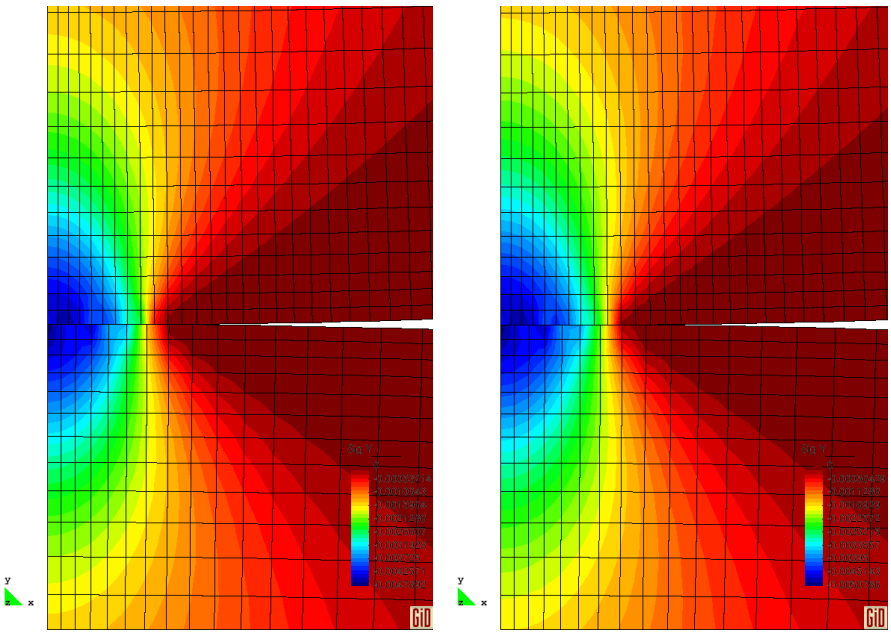


(e) 40 *s* **(f)** 50 *s*

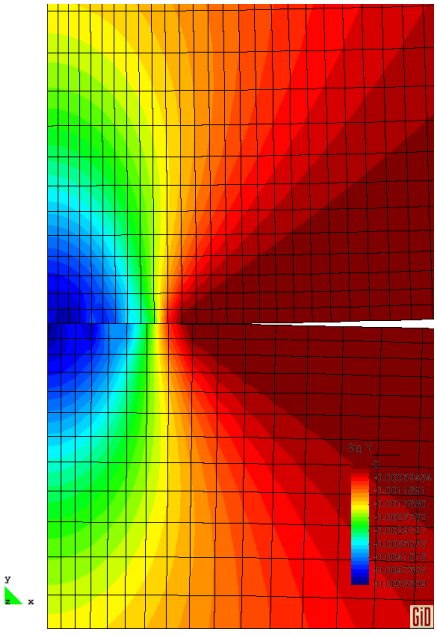


(g) 60 *s* **(h)** 70 *s*

Figure 6.13: Evolution of σ_y and contact status over pseudo time



(i) 80 *s* **(j)** 90 *s*



(k) 100 *s*

Figure 6.13: Evolution of σ_y and contact status over pseudo time

6.3.1.2 Discussion of results

First one should be aware, that the Hertzian assumptions are satisfied. The material law is isotropic linear elastic. The contact zone is small, which can be deduced from figure 6.14. Also compliance of the half space assumption can be derived from figure 6.14. No adhesion forces are modeled, therefore this assumption is

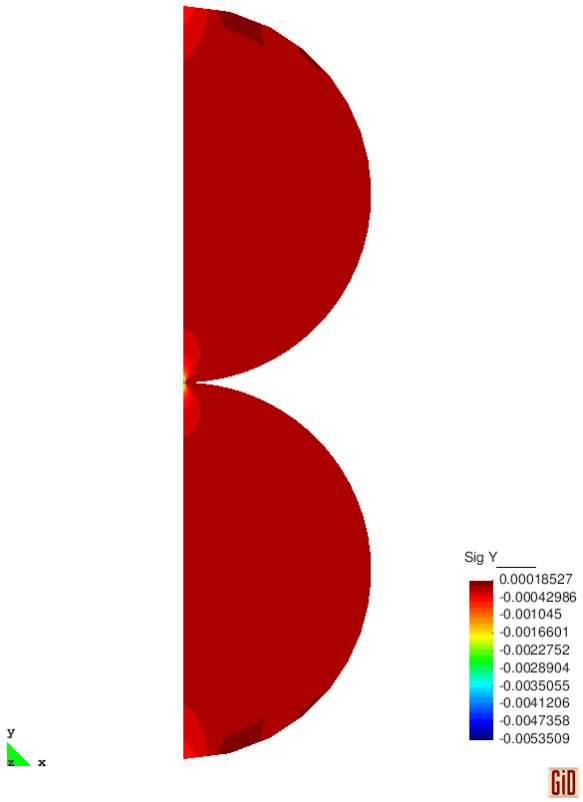


Figure 6.14: Stress σ_y and real displacement at 100 s

also satisfied. From chapter 4 we also know, that there are no frictional forces addressed in the implemented contact formulation. The infinite z-dimension is modeled by appropriate boundary conditions. Hence it is possible to compare the finite element solution to the analytical one from Hertz.

In figure 6.15 on the following page the normalized stress $-\sigma_y$ is plotted over the normalized distance. For the normalization the analytical values from table 6.4 on page 48 were taken.

For a fairly coarse mesh in the context of stress evaluation this is a good result. To be able to compare the solution with a state of the art finite element software package, Abaqus was selected. Within this software package the settings were set such that they are as close as possible to the Carat++ contact implementation. It was also a node-to-surface formulation picked with a penalty based enforcement method rule.

Due to the trick with the triangulated master surface and the non-matching meshes the problem is non-symmetric. That means that the force from the slave node to the master surface does neither act on the center of gravity of the mas-

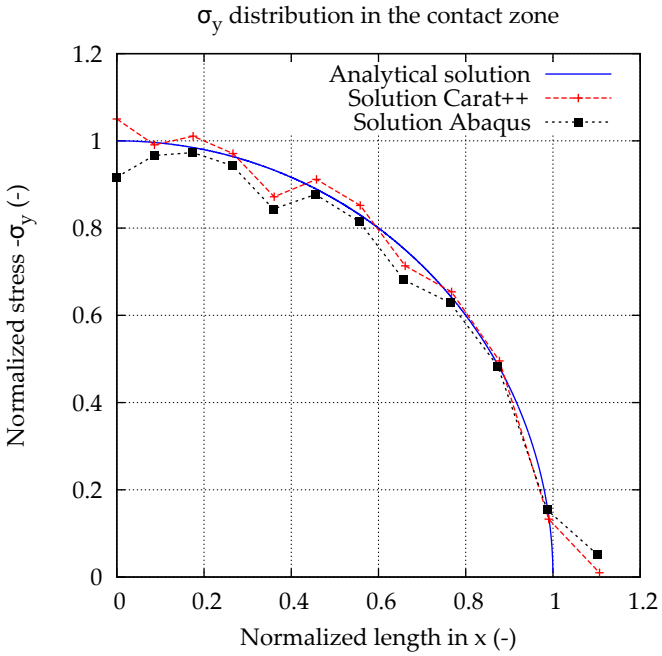


Figure 6.15: $\sigma_y(x)$ over x for analytical solution, Abaqus solution and Carat++ solution

ter faces nor at the corner nodes of the master faces. Therefore it is also worth to check the stress distribution in z-direction. From theory an constant stress is expected. Figure 6.16 equates to this exception.

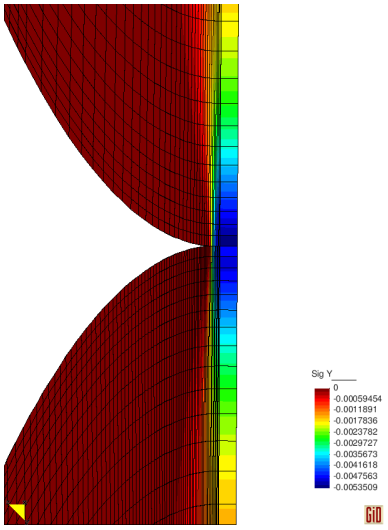


Figure 6.16: Stress σ_y in z-direction

The slight zigzagging of the stress in y-direction $\sigma_y(x)$ along x which is observed in figure 6.15, should be further examined. A detailed contour plot of this phenomena is shown in figure 6.17 on the following page. This is caused by slightly different slave node penetration distances due to the penalty method in combination with a non-linear solution method. Of course there are more reasons for this phenomena like round-off errors, the element formulation and the numerical integration. But investigations and comparisons with other contact for-

mulations in Abaqus show that the main reason are the input errors in the node coordinates and that the patch test is not fulfilled for a single master-slave pairing (see also section 6.2 on page 42).

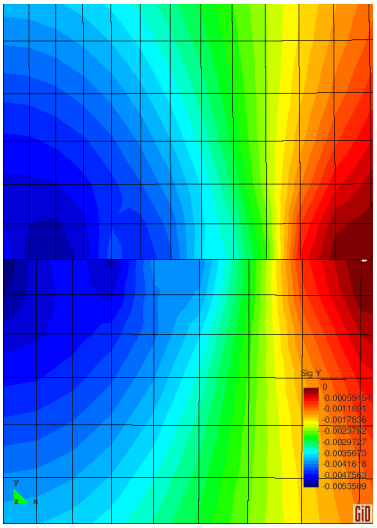


Figure 6.17: Stress $\sigma_y(x)$ close up

6.4 Sphere to egg morphing

The following test scenario was designed in order to illustrate the capabilities of the implemented contact algorithm. It is not intended to give deeper mechanical insight. A sphere is subjected to internal snow load. The sphere is clamped on three nodes lying on the equator. The sphere initially penetrates the floor. The floor can deform within the contact zone of the sphere and is clamped outside. The floor and the sphere are both modeled with triangular shell elements. The setting is summarized in figure 6.18.

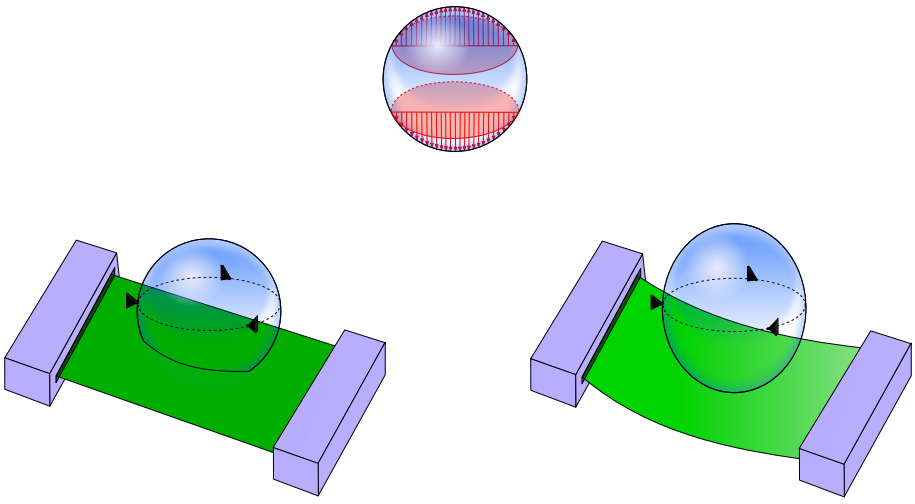
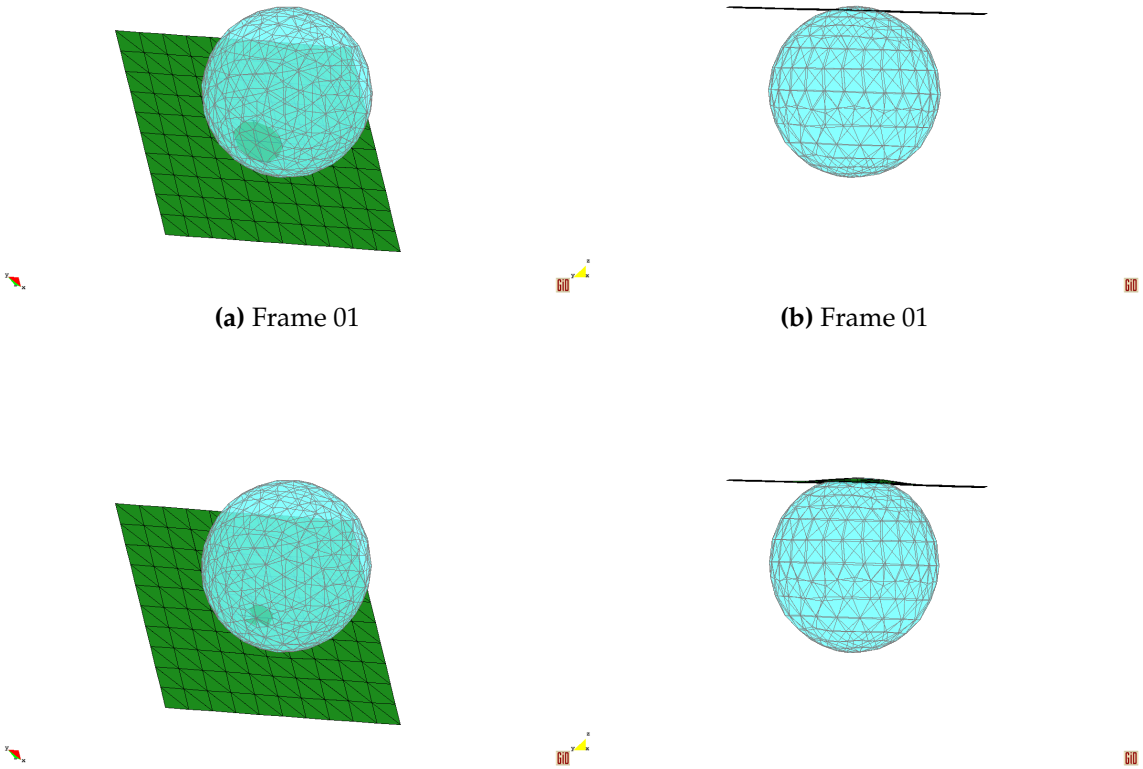


Figure 6.18: The setting of the "sphere to egg morphing" problem

Figure 6.19 shows the first two load steps of the scenario. The deformation plots of the other load steps and an additional view can be found in appendix B.1.



(a) Frame 01 **(b)** Frame 01

(c) Frame 02 **(d)** Frame 02

Figure 6.19: Sphere to egg morphing in two different views

6.4.1 Discussion of the deformation plots

Due to the coarse mesh there are moderate jumps in the normals of neighboring triangles. Although the contact searching approach is rather simple it works fine for this example, where the circumstances are not too easy. The problem was modeled without any symmetries. This causes a continuous change of the contact zone between sphere and floor. The basic implementation of contact can already represent quite complicated scenarios very well.

6.5 Concluding remarks

Although the contact algorithm is not highly sophisticated it gives very good results for the Hertzian contact example. With some tricks it is even possible to pass the contact patch test. It also handles examples with moderate large deformations (sphere to egg morphing) very well. With some of the enhancements

discussed in chapter 5 it could deal with a large number of problems. If one uses second order elements, one has to take care of the mid nodes. Because of the non-uniform element load vector, the mid nodes need a special treatment. One very simple approach is to constrain the mid node movement in the contact zone to the movement of the corner nodes.

The implementation process showed, that also in computational contact mechanics the object-oriented approach is a big advantage. Changes in the code are fairly simple to implement and the implementation effort for new features is less than in classical procedural programming.

Appendix

Appendix A

Mathematical preliminaries

A.1 Directional derivative

Let $f : \mathbb{R}^n \to \mathbb{R}$ where $f(x_1, ..., x_n)$ and $\Delta\mathbf{u} \in \mathbb{R}^n$ then the directional derivative is given by

$$\frac{\partial f(x_1, ..., x_n)}{\partial \Delta\mathbf{u}} = \nabla f(x_1, ..., x_n) \circ \Delta\mathbf{u} := \Delta f(x_1, ..., x_n) \tag{A.1}$$

A.2 Generalized chain rule

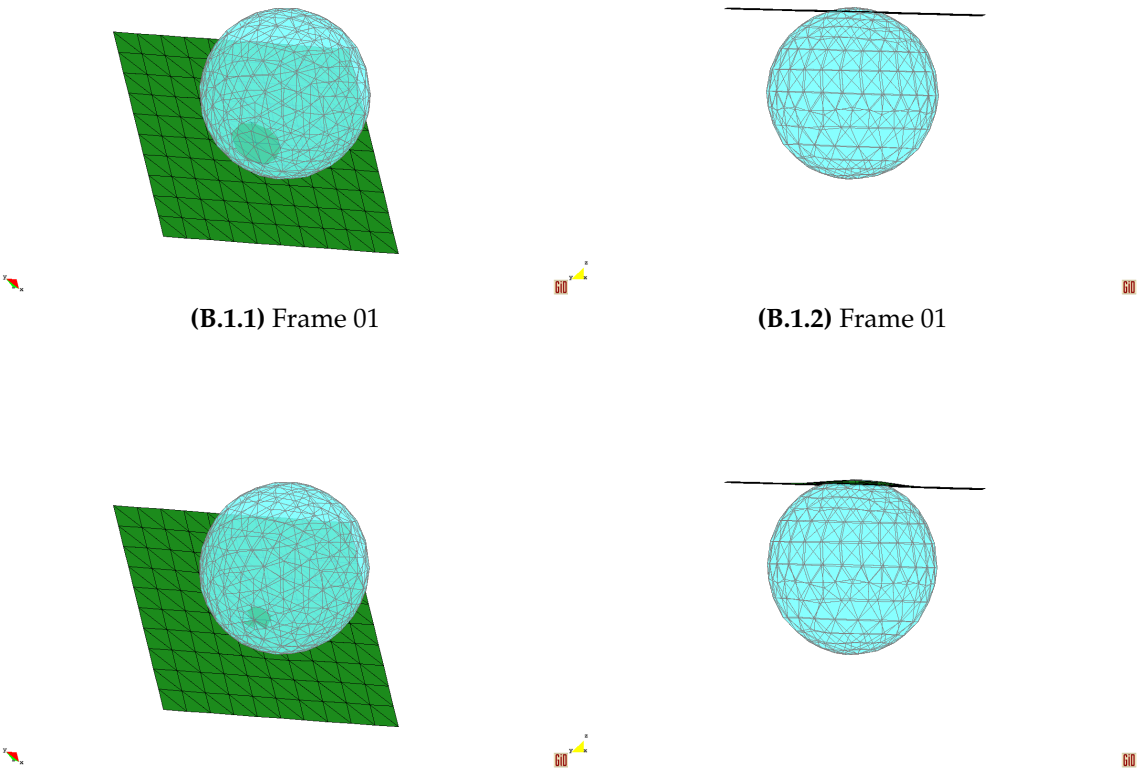
Let $f : \mathbb{R}^n \to \mathbb{R}$ and $g_i(t) : \mathbb{R} \to \mathbb{R} \; \forall \; i = 1, ..., n$ then the generalized chain rule gives the derivative of f with respect to t as

$$\frac{\mathrm{d}f(g_1(t), ..., g_n(t))}{\mathrm{d}t} = \sum_{k=1}^{n} \frac{\partial f(g_1(t), ..., g_n(t))}{\partial g_k} \frac{\mathrm{d}g_k(t)}{\mathrm{d}t} \tag{A.2}$$

Appendix B

Additional figures

B.1 Sphere to egg morphing



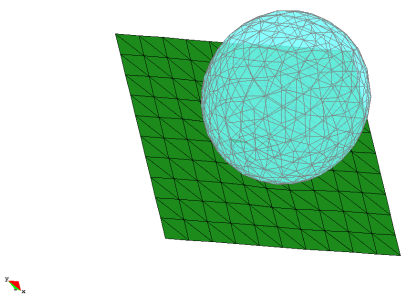
(B.1.1) Frame 01

(B.1.2) Frame 01

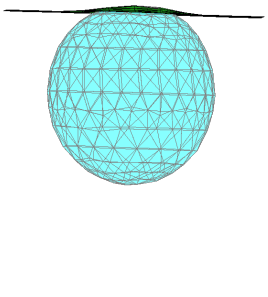
(B.1.3) Frame 02

(B.1.4) Frame 02

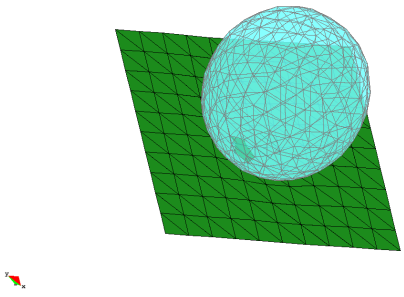
Figure B.1: Sphere to egg morphing in two different views



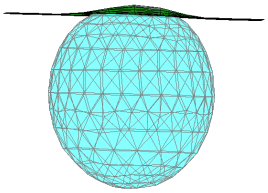
(B.1.5) Frame 03



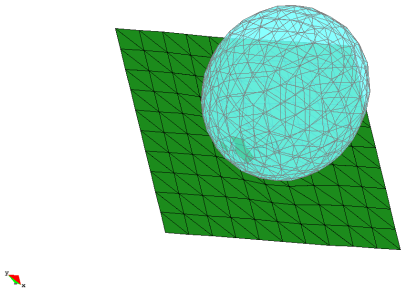
(B.1.6) Frame 03



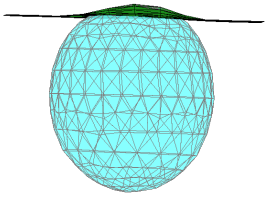
(B.1.7) Frame 04



(B.1.8) Frame 04

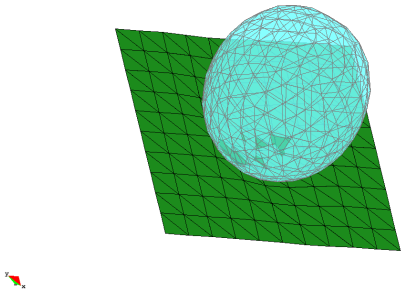


(B.1.9) Frame 05

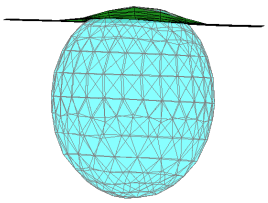


(B.1.10) Frame 05

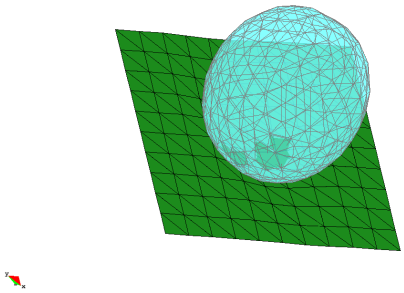
Figure B.1: Sphere to egg morphing in two different views



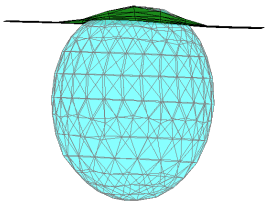
(B.1.11) Frame 06



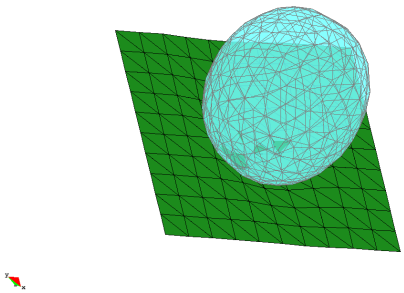
(B.1.12) Frame 06



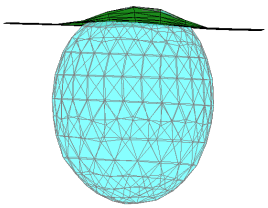
(B.1.13) Frame 07



(B.1.14) Frame 07

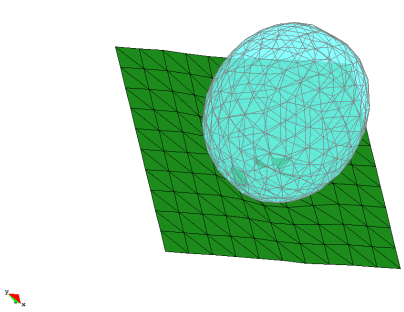


(B.1.15) Frame 08

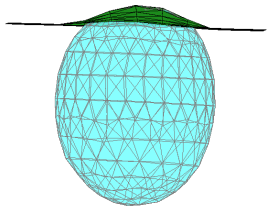


(B.1.16) Frame 08

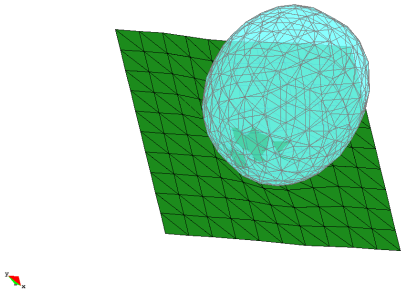
Figure B.1: Sphere to egg morphing in two different views



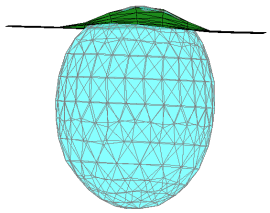
(B.1.17) Frame 09



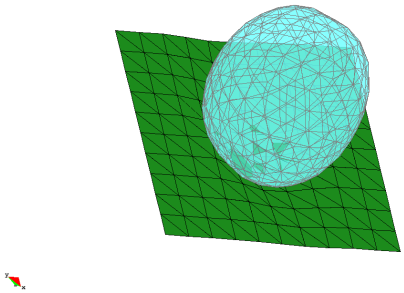
(B.1.18) Frame 09



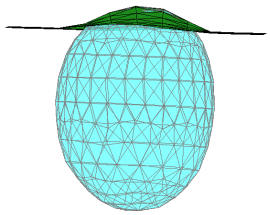
(B.1.19) Frame 10



(B.1.20) Frame 10

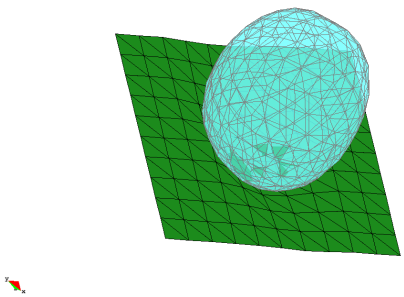


(B.1.21) Frame 11

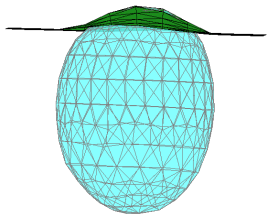


(B.1.22) Frame 11

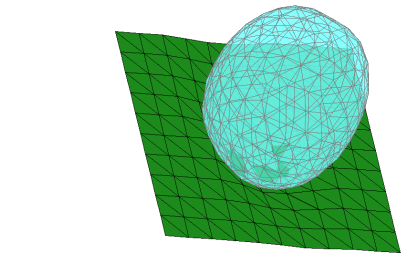
Figure B.1: Sphere to egg morphing in two different views



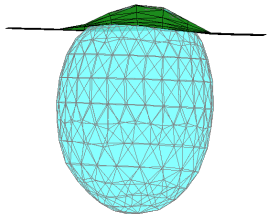
(B.1.23) Frame 12



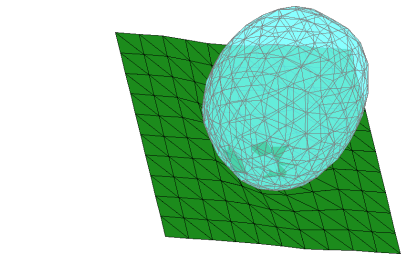
(B.1.24) Frame 12



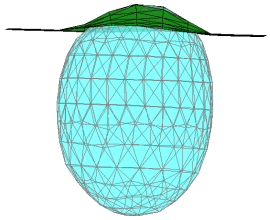
(B.1.25) Frame 13



(B.1.26) Frame 13

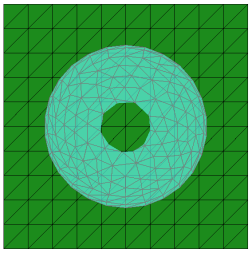


(B.1.27) Frame 14

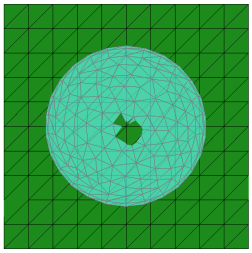


(B.1.28) Frame 14

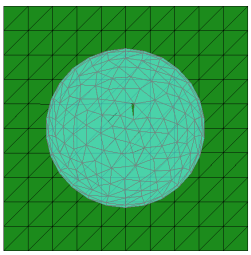
Figure B.1: Sphere to egg morphing in two different views



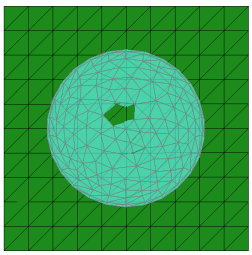
(B.2.1) Frame 01



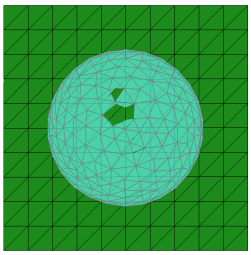
(B.2.2) Frame 02



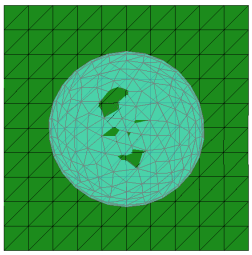
(B.2.3) Frame 03



(B.2.4) Frame 04

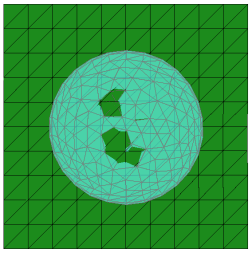


(B.2.5) Frame 05

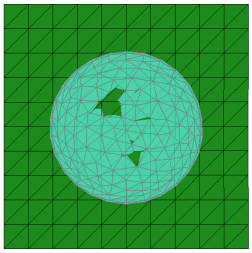


(B.2.6) Frame 06

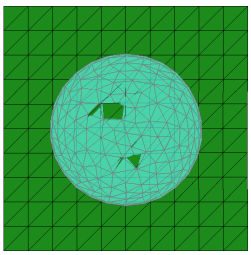
Figure B.2: Sphere to egg morphing (view: sphere cut at the equator)



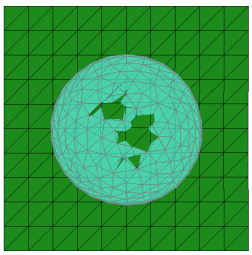
(B.2.7) Frame 07



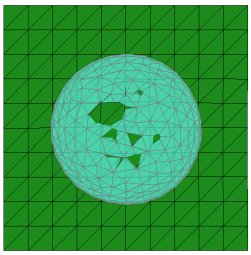
(B.2.8) Frame 08



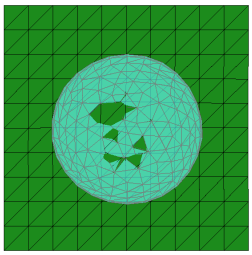
(B.2.9) Frame 09



(B.2.10) Frame 10

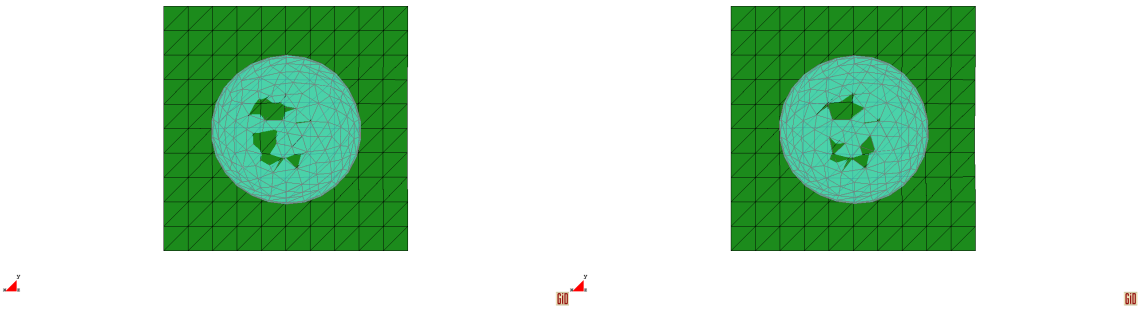


(B.2.11) Frame 11



(B.2.12) Frame 12

Figure B.2: Sphere to egg morphing (view: sphere cut at the equator)



(B.2.13) Frame 13 **(B.2.14)** Frame 14

Figure B.2: Sphere to egg morphing (view: sphere cut at the equator)

Appendix C

Shell versus truss example

Input file C.1: One degree of freedom example

```
!########################################################################
!####                              PC-BLOCK                            ####
!########################################################################
! Truss vs Shell
!=======================================================================
PC-PROBLEM
  MASTERJOB = PC-ANALYSIS 1
!=======================================================================

!=======================================================================
PC-ANALYSIS 1: STA_GEO_NONLIN
  PATHCONTROL = FORCE ! or DISPLACEMENT or ARCLENGTH
  SOLVER = PC-SOLVER 5
  OUTPUT = PC-OUT 1
  COMPCASE = LD-COM 1
  DOMAIN = EL-DOMAIN 1
  NUM_STEP = 1
  MAX_ITER_EQUILIBRIUM = 20
  EQUILIBRIUM_ACCURACY = 1e-08
  CURVE = LD-CURVE 1
  TRACED_NODE = 4
  TRACED_NODAL_DOF = DISP_Z
  CONTACT = CONTACT 1
!=======================================================================

!=======================================================================
PC-SOLVER 5: CROUT_SKYLINE
 BANDWITH = CUTHILL_MCKEE
!=======================================================================

!=======================================================================
PC-OUT 1 : GID
   GEOM=1
   DISP=1
   SUPPORT_FORCE=1
   STRESS=1
!=======================================================================
!########################################################################
!####                              ND-BLOCK                            ####
!########################################################################
!=======================================================================
!=====================================================================
ND-COOR
     NODE   1 X    0.0000000 Y    1.0000000 Z    0.0000000
     NODE   2 X   -0.8660000 Y   -0.5000000 Z    0.0000000
     NODE   3 X    0.8660000 Y   -0.5000000 Z    0.0000000
     NODE   4 X    0.0000000 Y    0.0000000 Z   -0.0010000
     NODE   5 X    0.0000000 Y    0.0000000 Z    1.0000000
!=====================================================================
```

```
!=================================================================
BC-DIRICHLET 1: SPC-ZERO
 NODE         2  DISP_X, DISP_Y, DISP_Z, DIR_DIFF_SHELL8_X, DIR_DIFF_SHELL8_Y,
     DIR_DIFF_SHELL8_Z
 NODE         3  DISP_X, DISP_Y, DISP_Z, DIR_DIFF_SHELL8_X, DIR_DIFF_SHELL8_Y,
     DIR_DIFF_SHELL8_Z
 NODE         4  DISP_X, DISP_Y
 NODE         5  DISP_X, DISP_Y, DISP_Z
!=================================================================
!#################################################################
!####                        CONTACT-BLOCK                    ####
!#################################################################
!=================================================================
SURFACE 2 : NODE
4
!=================================================================

!=================================================================
SURFACE 1 : ELEMENT
1, S1
!=================================================================

!=================================================================
SURFACE-INTERACTION 1
PHYSICAL-NORMAL-BEHAVIOR=HARD
PHYSICAL-TANGENTIAL-BEHAVIOR=NOFRICTION
CONSTRAINT-ENFORCEMENT-NORMAL=PENALTY_CONST
PENALTY_PARAMETER=1.0
!=================================================================

!=================================================================
CONTACT-PAIR 1
SLAVE-SURFACE = SURFACE 2
MASTER-SURFACE = SURFACE 1
CONTACT-PROPERTIES = SURFACE-INTERACTION 1
CONTACT-TRACKING = SMALL-SLIDING !FINITE-SLIDING
CONTACT-DISCRETIZATION = NODE-TO-SURFACE
!=================================================================

!=================================================================
CONTACT 1
CONTACT-PAIRS = CONTACT-PAIR 1
!=================================================================


!#################################################################
!####                          EL-BLOCK                       ####
!#################################################################
!=================================================================
EL-PART 1 NAME=Shell
!=================================================================

!=================================================================
EL-MAT 1 : LIN_ELAST_ISOTROPIC
 EMOD=1 ALPHAT=0.0  DENS=7810 NUE=0.0 XSI=1.2
!=================================================================

!=================================================================
EL-PROP 1 : SHELL8
MAT = EL-MAT 1
THICKNESS= 0.1
INT_TYPE_SHELL8 = FULL
SDC= 1.0
EAS = 0,0,0,0,0
ANS = NONE
FORCE = RST_ORTHO
!=================================================================

!=================================================================
```

```
EL-PART 2 NAME=truss
!=====================================================================

!=====================================================================
EL-PROP 2 : TRUSS1
MAT= EL-MAT 1          AREA=1.0
PRESTRESS         SIG11=0.0
LAGRANGE=TOTAL
!=====================================================================

!=====================================================================
EL-TOP 1
!      EL_ID   PART   PROP   NODE1   NODE2 ....
 NEL      1     1      1     1       2        3
 NEL      2     2      2     4       5
!=====================================================================

!=====================================================================
EL-DOMAIN 1
 ELEMENTS = EL-TOP 1
!=====================================================================

!#####################################################################
!####                              LD-BLOCK                         ####
!#####################################################################

!=====================================================================
LD-CURVE 1 TYPE=DISCRETE
 TIME=0.000    VAL=0.000
 TIME=1.000    VAL=1.000
!=====================================================================

!=====================================================================
 LD-COM 1
 TYPE=BC-DIRICHLET 1
!=====================================================================
```

For the first iteration of the problem described in the input file above the tangent stiffness matrix reads:

$$\begin{bmatrix}
0.00000 & 0.00000 & 0.00000 & 0.00000 & 0.00000 & 0.00000 & 0.00000 & 0.00000 & 0.00058 & 0.00000 & 0.00000 & -0.00058 \\
0.00000 & 0.00000 & 0.00000 & 0.00000 & 0.00000 & -0.00067 & 0.00000 & 0.00000 & 0.00033 & 0.00000 & 0.00000 & 0.00033 \\
0.00000 & 0.00000 & 1.00000 & 0.00000 & 0.00000 & -0.33333 & 0.00000 & 0.00000 & -0.33333 & 0.00000 & 0.00000 & -0.33333 \\
0.00000 & 0.00000 & 0.00000 & 0.00000 & 0.00000 & 0.00000 & 0.00000 & 0.00000 & -0.00019 & 0.00000 & 0.00000 & 0.00019 \\
0.00000 & 0.00000 & 0.00000 & 0.00000 & 0.00000 & 0.00022 & 0.00000 & 0.00000 & -0.00011 & 0.00000 & 0.00000 & -0.00011 \\
0.00000 & -0.00067 & -0.33333 & 0.00000 & 0.00022 & 0.11111 & 0.00000 & 0.00022 & 0.11111 & 0.00000 & 0.00022 & 0.11111 \\
0.00000 & 0.00000 & 0.00000 & 0.00000 & 0.00000 & 0.00000 & 0.00000 & 0.00000 & -0.00019 & 0.00000 & 0.00000 & 0.00019 \\
0.00000 & 0.00000 & 0.00000 & 0.00000 & 0.00000 & 0.00022 & 0.00000 & 0.00000 & -0.00011 & 0.00000 & 0.00000 & -0.00011 \\
0.00058 & 0.00033 & -0.33333 & -0.00019 & -0.00011 & 0.11111 & -0.00019 & -0.00011 & 0.11111 & -0.00019 & -0.00011 & 0.11111 \\
0.00000 & 0.00000 & 0.00000 & 0.00000 & 0.00000 & 0.00000 & 0.00000 & 0.00000 & -0.00019 & 0.00000 & 0.00000 & 0.00019 \\
0.00000 & 0.00000 & 0.00000 & 0.00000 & 0.00000 & 0.00022 & 0.00000 & 0.00000 & -0.00011 & 0.00000 & 0.00000 & -0.00011 \\
-0.00058 & 0.00033 & -0.33333 & 0.00019 & -0.00011 & 0.11111 & 0.00019 & -0.00011 & 0.11111 & 0.00019 & -0.00011 & 0.11111
\end{bmatrix}$$

The residual vector reads:

$$\begin{bmatrix}
-0.00000 \\
-0.00000 \\
-0.00100 \\
0.00000 \\
0.00000 \\
0.00033 \\
0.00000 \\
0.00000 \\
0.00033 \\
0.00000 \\
0.00000 \\
0.00033
\end{bmatrix}$$

Nomenclature

Greek letters

α	index, where $\alpha = \{1, 2\}$
β	index, where $\beta = \{1, 2\}$
χ	index, where $\chi = \{1, 2\}$
ϵ	strain
γ	index, where $\gamma = \{1, 2\}$
ι	real number
$\phi_i(x)$	shape function of node i
Π	quadratic functional
π	≈ 3.141593
ρ	density
σ	stress
ξ_α	convective coordinates of master face

Mathematical symbols

Σ	sum
∂	operator for partial derivative
Δ	directional derivative
δ	first variation
$\mathbb{R}$	set of real numbers
$\circ$	scalar product

Latin letters

A	cross sectional area in the deformed configuration

a	acceleration
A_0	cross sectional area in the initial configuration
$\bar{a}^{\alpha\beta}$	contravariant metric
$\bar{a}_{\alpha\beta}$	covariant metric
$\bar{\mathbf{a}}_{\alpha}^{M}$	covariant tangent vector of the master face
$\bar{\mathbf{a}}^{M\alpha}$	contravariant tangent vector of the master face
A_n	contact area associated to slave node n
b	minimal distance between master and slave surface
c	penalty stiffness
$\bar{\mathbf{x}}_{,\alpha}^{M}$	derivative of $\bar{\mathbf{x}}^{M}$ with respect to the convective coordinates ξ_α
E	Young's modulus
$\mathbf{f}$	global external force vector
$\mathbf{f}^{ei}$	element external force vector of element i
g	gap function
$\mathbf{K}$	global stiffness matrix
$\mathbf{K}_T$	global tangent stiffness matrix
$\mathbf{K}^{ei}$	element stiffness matrix of element i
L	length in the initial configuration
l	length in the deformed configuration
n	line load
$N(x)$	normal force as function of x
$\bar{\mathbf{n}}^{M}$	unit normal of the master face
n_c	number of slave nodes
O	linear operator
$\mathbf{p}$	global internal force vector
r	radius
S	trial space

$\Delta\bar{\mathbf{u}}^M$ displacement increments for the projection of the slave node onto the master face

$\Delta\mathbf{u}^S$ displacement increments for the slave node

$\mathbf{r}$ global residual vector

$\mathbf{u}$ vector of displacements

$u(x)$ displacement in x-direction as function of x

V volume

W test space

$w(x)$ weighting function

$\bar{\mathbf{x}}^M$ coordinates of the projection of the slave node onto the master face

$\mathbf{x}_i^M$ coordinates of node i of the master face

$\mathbf{x}^S$ coordinates of the salve node

Abbreviations:

BMW Bayerische Motorenwerke

FEM finite element method

FSI fluid-structure interaction

TFSI thermal fluid-structure interaction

List of Figures

List of Tables

Bibliography

[1] N. El-Abbasi and K.J. Bathe.
"Stability and patch test performance of contact discretizations and a new solution algorithm."
In: *Computers & Structures* 79.16 (2001), pp. 1473–1486.

[2] R. Aris.
Vectors, Tensors and the Basic Equations of Fluid Mechanics.
Dover Publications Inc, 1989.

[3] G. Bärwolff.
Höhere Mathematik für Naturwissenschaftler und Ingenieure.
second.
Spektrum Akademischer Verlag, 2006.

[4] W. Beitz and K.-H. Grote.
DUBBEL Taschenbuch für den Maschinenbau.
20.
Teubner, 2001.

[5] U. Breymann.
C++: Einführung und professionelle Programmierung.
Hanser Verlag, 2007.

[6] I.N. Bronstein, K.A. Semendjajew, and G. Musiol.
Taschenbuch der Mathematik.
Harri Deutsch Verlag, 2008.

[7] P. Chadwick.
Continuum Mechanics: Concise Theory and Problems.
Dover Publications Inc, 1998.

[8] L. D. Elsgolc.
Calculus of Variations.
Dover Publications Inc, 2007.

[9] I. M. Gelfand and S. V. Fomin.
Calculus of Variations.
Dover Publications Inc, 2000.

[10] H. Haf and F. Wille.
Höhere Mathematik für Ingenieure Band IV Vektoranalysis und Funktionentheorie.
Teubner Verlag, 1990.

[11] H. Hertz.
"Über die Berührung fester elastischer Körper."
In: *Journal für die reine und angewandte Mathematik* 92 (1881), pp. 156–171.

[12] R. Hill.
"Aspects of Invariance in Solid Mechanics."
In: *Advances in applied mechanics* 18 (1978), pp. 1–75.

[13] G. A. Holzapfel.
Nonlinear Solid Mechanics: A Continuum Approach for Engineering.
John Wiley & Sons, 2000.

[14] T. J. Hughes, R. L. Taylor, and W. Kanoknukulchai.
"A finite element method for large displacement contact and impact problems."
In: *Formulations and Computational Algorithms in Finite Element Analysis*.
Ed. by Klaus-Jürgen Bathe, J. Tinsley Oden, and Walter Wunderlich.
MIT Press, 1977,
Pp. 468–496.

[15] N. Kikuchi and J. T. Oden.
Contact Problems in Elasticity: A Study of Variational Inequalities and Finite Element Methods.
SIAM, 1988.

[16] T. A. Laursen.
Computational Contact and Impact Mechanics: Fundamentals of Modeling Interfacial Phenomena in Nonlinear Finite Element Analysis.
Springer Verlag, 2002.

[17] L. E. Malvern.
Introduction to the Mechanics of a Continuous Medium.
Prentice Hall, 1969.

[18] J. E. Marsden and T. J. Hughes.
Mathematical Foundations of Elasticity.
Dover Publications Inc, 1994.

[19] L. Nasdala.
FEM-Formelsammlung Statik und Dynamik: Hintergrundinformationen, Tipps und Tricks.
Vieweg+Teubner Verlag, 2010.

[20] B. Nour-Omid and P. Wriggers.
"A note on the optimum choice for penalty parameters."
In: *Communications in Applied Numerical Methods* 3.6 (1987), pp. 581–585.

[21] R. W. Ogden.
Non-Linear Elastic Deformations.
Dover Publications Inc, 1997.

[22] SIMULIA.
Abaqus Analysis User´s Manual Volume I-V.
v6.10.
2010.

[23] SIMULIA.
Abaqus Theory Manual.
v6.10.
2010.

[24] G. Strang.
Computational Science and Engineering.
Wellesley-Cambridge Press, 2007.

[25] G. Strang and G. Fix.
An Analysis of the Finite Element Method.
Wellesley-Cambridge Press, 2008.

[26] M. J. Turner et al.
"Stiffness and deflection analysis of complex structures."
In: *J. Aeronautical Society* 23 (1956).

[27] T. Westermann.
Mathematik für Ingenieure mit Maple Band 1.
4th.
Springer Verlag, 2004.

[28] T. Westermann.
Mathematik für Ingenieure mit Maple Band 2.
second.
Springer Verlag, 2001.

[29] Wikipedia.
Wikipedia, The Free Encyclopedia.
`http://en.wikipedia.org/`.
2010.
(Visited on 08/22/2010).

[30] J.R. Williams and R. O'Connor.
"Discrete element simulation and the contact problem."
In: *Archives of Computational Methods in Engineering* 6.4 (1999), pp. 279–304.

[31] P. Wriggers.
Computational contact mechanics.
second.
Springer Verlag, 2006.

[32] P. Wriggers.
Konsistente Linearisierungen in der Kontinnumsmechanik und ihre Anwendung auf die Finite-element-methode.
Inst. für Baumechanik u. Numerische Mechanik Universität Hannover, 1988.

[33] P. Wriggers.
Nichtlineare Finite-Element-Methoden.
Springer Verlag, 2001.

[34] B. Yang, T.A. Laursen, and X. Meng.
"Two dimensional mortar contact methods for large deformation frictional sliding."
In: *International Journal for Numerical Methods in Engineering* 62.9 (2005), pp. 1183–1225.

[35] D. Yang.
C++ and object-oriented numeric computing for scientists and engineers.
Springer Verlag, 2001.

[36] G. Zavarise and L. De Lorenzis.
"A modified node-to-segment algorithm passing the contact patch test."
In: *International Journal for Numerical Methods in Engineering* 79.4 (2009), pp. 379–416.

[37] G. Zavarise and L. De Lorenzis.
"The node-to-segment algorithm for 2D frictionless contact: Classical formulation and special cases."
In: *Computer methods in applied mechanics and engineering* 198.41-44 (2009), pp. 3428–3451.

[38] O. C. Zienkiewicz and K. Morgan.
Finite Elements & Approximation.
Dover Publications Inc, 2006.

[39] O. C. Zienkiewicz and R. L. Taylor.
The Finite Element Method for Solid and Structural Mechanics.
6th.
Butterworth Heinemann, 2005.

[40] O. C. Zienkiewicz, R. L. Taylor, and J. Z. Zhu.
The Finite Element Method. Its Basis and Fundamentals.
6th.
Butterworth Heinemann, 2005.