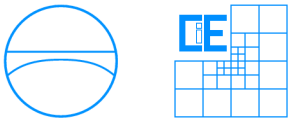


Faculty of Civil, Geo and Environmental Engineering
Chair for Computation in Engineering
Prof. Dr. rer. nat. Ernst Rank

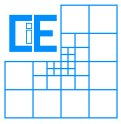
Large deformation two- and three-dimensional contact on embedded interfaces using the Finite Cell Method

Alexandre Mongeau

Master's thesis
for the Master of Science program Computational Mechanics

Author:	Alexandre Mongeau
Supervisor:	Prof. Dr. rer. nat. Ernst Rank
	Dipl.-Ing. Tino Bog
Date of issue:	18. June 2015
Date of submission:	18. December 2015

Involved Organisations



Chair for Computation in Engineering
Faculty of Civil, Geo and Environmental Engineering
Technische Universität München
Arcisstraße 21
D-80333 München

Declaration

With this statement I declare, that I have independently completed this Master's thesis. The thoughts taken directly or indirectly from external sources are properly marked as such. This thesis was not previously submitted to another academic institution and has also not yet been published.

München, December 16, 2015

Alexandre Mongeau

Alexandre Mongeau
Oskar-von-Miller-Ring 25
D-80333 München
e-Mail: alexandre.mongeau@tum.de

Acknowledgments

I would like to acknowledge and express my sincere thanks to the many people who have made the realization of this work possible.

First of all, I would like to express my gratitude to my supervisor Dipl.-Ing Tino Bog, who from the beginning has shown great interest in my project. His invaluable advice and support were keys to guide this work to completion. His positive attitude and composed yet categorical approach to new challenges and difficulties were greatly appreciated.

I would like to thank Prof. Dr. rer. nat. Ernst Rank for giving me the opportunity to work on this very interesting topic and accepting to take part in the evaluation of the present work.

I would also like to take this opportunity to thank my colleagues Massimo, Luca and Davide for the technical discussions as well as the friendly atmosphere in the working room of the Chair for Computation in Engineering.

My thanks also go to Céline, François, Azadeh and the rest of my family and friends for their support and help during the long hours of writing.

München, December 16, 2015 Alexandre Mongeau

Nomenclature

$\mathbf{A}$	Shape function matrix
$\mathbf{A}_{,i}$	Shape function matrix derived w.r.t. master local coordinates
a^{ij}	Contravariant metric tensor
$\mathbf{F}$	Deformation gradient
f	Volumetric load
G	Shear Modulus
g	Gap value
h_{ij}	Covariant curvature tensor
J	Jacobian
$\mathbf{n}$	Normal on the master surface
$\mathbf{r}$	Slave point vector
$\mathbf{u}$	Global solution vector (displacements)
$\delta\mathbf{u}$	Virtual displacement vector
$\mathbf{v}_{ma}$	Master point tangential velocity
$\mathbf{v}_{sl}$	Master point tangential velocity
$\mathbf{v}$	Relative velocity vector
$\mathbf{v}_{,i}$	Derived relative velocity vector
v_t	Relative tangential velocity
N	Normal force
N_{ma}	Master body shape functions
$N_{ma,i}$	Derived master body shape functions
N_{sl}	Slave body shape functions
$\mathbf{R}^N$	Normal contact residual
$\mathbf{R}^T$	Tangential contact residual
S_{ma}	Master surface
S_{sl}	Slave surface

$\mathbf{T}_{ma}$	Total master traction vector
$\mathbf{T}_{sl}$	Total slave traction vector
T_i	Covariant components of the traction vector
T_i^{el}	Covariant components of the elastic trial traction vector
W_C	Total contact virtual work
W_C^N	Normal contact virtual work
W_C^T	Tangential contact virtual work
$W_C^{T,st}$	Tangential sticking contact virtual work
$W_C^{T,sl}$	Tangential sliding contact virtual work
$\mathbf{X}$	Reference configuration
$\mathbf{x}$	Current configuration
$\delta\mathbf{x}$	Relative virtual displacement vector
$\Delta\boldsymbol{\rho}$	Tangential gap
δ_i^j	Kronecker delta
α	Indicator function
β	Penalty parameter for the weak enforcement of constraints
η_i	Convective coordinates of the slave surface
ϵ_{ij}	Linear strain tensor
ϵ_N	Normal penalty parameter
ϵ_T	Tangential penalty parameter
Γ	Boundary of the slave or master body
Γ_c	Part of the boundary subject to contact constraint conditions
Γ_D	Part of the boundary subject to Dirichlet boundary conditions
Γ_N	Part of the boundary subject to Neumann boundary conditions
Γ_{ij}^k	Christoffel symbols
μ	Coefficient of friction
Ω	Deformable body
ω_i	Gaussian quadrature weights
Φ	Trial yield function
$\boldsymbol{\rho}$	Master point vector
$\boldsymbol{\rho}_i$	Tangent vectors on the master surface
σ_{ij}	Cauchy stress tensor
ξ^i	Convective coordinates of the master surface
$\dot{\xi}^i$	Rate of deformation in convective coordinates

Contents

Chapter 1

Introduction

The extraordinary fast pace at which computer technology has been evolving over the recent years allows the simulation of large and complex problems which would have been deemed unrealistic in the past. In many cases, real world problems of interest involve contact between different parts. Common machining processes such as rolling, deep drawing as well as gear assemblies and bearings, which are found in most mechanical systems, are some of the few examples. The most known and widespread method for the simulation of such problems is the finite element method. There is a vast variety of software packages available on the market which include this method. Recently, a lot of research has been done on the development of high-order finite element methods. These use higher-order polynomials for the interpolation of the quantities of interest as opposed to the linear polynomials used in the classical finite element method. This new version brought many advantages such as a better approximation of the geometry and a higher accuracy of the solution.

As such methods grew in popularity, so did the interest in reducing the overall analysis time. It was noted that the greatest impediment in the complete simulation procedure did not reside in the actual solving of the problem, but in the pre-processing step. This step can take up to 80% of the time required for the simulation, which corresponds to approximately 60% of the time for geometry creation and 20% for the generation of a conforming mesh, according to the data obtained by Cottrell et al. (2009) of a study concerning Sandia National Laboratories. Improved formulations can therefore result in a significant and immediate improvement to the cost and time of generating accurate solutions.

In order to circumvent the time-consuming pre-processing step, embedded interface methods, which require no meshing of the studied bodies, made their appearance. The finite cell method, developed at the Chair for Computation in Engineering of the Technische Universität München (Technical University of Munich) is an example of such methods. It was first introduced by Parvizian et al. (2007). It combines high-order finite elements with a special integration technique to analyze structures of arbitrary shapes. This method has been extensively developed in the last decade. Its applicability to the analysis of problems involving large deformations, which are from a complexity point of view far more arduous than small deformation problems, has also been demonstrated by Schillinger et al. (2012).

However, one of the remaining topics is the simulation of contact. Computational contact mechanics is a particularly complicated branch of computational mechanics as the contact

constraints are highly non-linear. A variety of different formulations have been proposed for the finite element method although very few have successfully been applied to the finite cell method, see Konyukhov et al. (2015). For this reason, there is a need for a robust, multi-purpose simulation algorithm for contact problems in a finite cell method framework.

AdhoC++, the C++ based framework developed at the Chair for Computation in Engineering is a state-of-the-art high-order finite element code which includes finite cell method analysis capabilities. For researchers, it serves as a testing ground for the development and implementation of new algorithms for a variety of different engineering applications. Although a few methods for contact were already implemented, they mainly focused on the analysis of an elastic body coming in contact with a rigid surface.

The main objective of this work is to research, analyze and compare different works on the penalty regularization of contact constraints which could be applicable to an embedded interface method such as the finite cell method. Amongst the requirements are the possibility to study arbitrarily shaped bodies, the applicability to self-contact and large deformations. The manuscript by Schillinger and Ruess (2014) shows just how versatile and efficient the finite cell method can be. It is proven therein how this method can also be used in the case of large deformations. However, the combination of contact mechanics, large deformations and the FCM are still a highly topical and promising subject.

The present work focuses on numerical treatment of contact problems for the finite cell method. The basic concepts of this method along with some relevant particularities are given in chapter 2. Afterwards, an overview of the field of computational contact mechanics and the general formulation of the problem is presented in chapter 3. The central point of the current work follows, which consists of three different steps:

1. Global search
2. Local search
3. Constraint enforcement

The *global search* routine is first run in order to reduce the overall computation costs engendered by the mathematical model used for the contact constraints. This is covered in chapter 4. Moreover, contact constraints need to be applied only the intersecting sections of the studied bodies. The *local search* is run after the global search and provides the precise contacting areas of the two bodies through a closest-point projection process. It is discussed in chapter 5. Once the non-contacting parts have been filtered out, the contact boundary conditions can be computed and applied on the remaining elements. Contact can be separated into two categories. If only the normal forces are considered, it is referred to as *normal* contact or *frictionless* contact. If the tangential forces are considered in the analysis, a different formulation needs to be used. It is referred to as *tangential* contact or *frictional* contact. These are the topic of chapter 6. Following this theoretical part, some applications based on what was previously discussed are presented in chapter 7, accompanied by the numerical results obtained from the AdhoC++ framework. Lastly, a summary of the results and some general comments and observations are made in chapter 8.

Chapter 2

The Finite Cell Method

The field of computational mechanics, even though it made its appearance decades ago, is still evolving constantly. Many engineering problems can be modeled using partial differential equations (PDE), see Zienkiewicz and Taylor (1977). Most PDEs have an infinite amount of solutions, or none at all. The objective is to obtain a well-posed PDE. By definition, this means to have a unique solution which depends continuously on the problem data. To do so, some boundary conditions can be specified which the solution needs to respect on every point of the boundary while the PDE holds for the interior of the domain. These are referred to as boundary value problems.

Once the boundary value problem is well-defined, many approaches were developed to solve it numerically. The finite element method was first introduced in the early 1960s and it now a very popular and well-known technique used in most engineering analysis and design processes. The numerical solution resulting from the application of this method has allowed engineers and physicists of a variety of branches to gain invaluable insight on the physics behind otherwise analytically unsolvable problems.

Even if the finite element method has revolutionized the engineering world, research is still ongoing on topics leading to possible improvements. It also lead to the introduction of new methods, one of which is known as the finite cell method. This approach removes the need for the meshing step of the classical FEM, drastically reducing the time required for pre-processing.

The first part of this chapter serves as an introduction to the finite element method. An overview of the theory of linear elasticity is presented, followed by the extension to high-order finite elements. Afterwards, the concepts behind the finite cell method are explained. Some extensions to the FCM are then discussed and a technique to recover geometries in an embedded domain, which proves very useful in the current work, is covered.

2.1 The Finite Element Method - Basic Principles

The basic idea behind the finite element method is to divide the geometry into individual components, called finite elements. The behavior of these elements is simple and can be described using simple mathematical equations. The interaction between the individual com-

ponents is considered during the assembly process, where the elements are joined to obtain an approximation of the response of the global system. The concepts and equations presented here are an overview of the finite element method. The books of Bathe (1996) and Hughes (2000) provide more thorough and detailed information.

The present section focuses on the presentation of the finite element method as an introduction to the finite cell method.

2.1.1 Theory of Linear Elasticity

The theory of elasticity restricts itself to the description of the elastic deformation of materials. Therefore, the equations developed herein are limited to the modeling small deformation problems. The derivations presented here follow closely Zienkiewicz and Taylor (2005) and Franke (2011).

2.1.1.1 Kinematics

The first equations are obtained directly from a geometrical analysis. Kinematics study the movement of bodies in space, from an initial or reference point $\mathbf{X}$ to a deformed or current point $\mathbf{x}$. These can be expressed in any basis. The most commonly used is the Cartesian reference frame:

$$\begin{aligned} \mathbf{X} &= X^1\mathbf{e_1} + X^2\mathbf{e_2} + X^3\mathbf{e_3} \\ \mathbf{x} &= x^1\mathbf{e_1} + x^2\mathbf{e_2} + x^3\mathbf{e_3} \end{aligned} \tag{2.1}$$

The relationship between the initial and current configuration involves the displacement vector:

$$\mathbf{u} = \mathbf{x} - \mathbf{X} \tag{2.2}$$

The deformation can be expressed using the strain, which is directly related to this displacement vector:

$$\epsilon_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) \tag{2.3}$$

It is worth mentioning that equation 2.3 is only valid for small deformations since some simplifying assumptions were used to obtain this form of the equation (Zienkiewicz and Taylor (1977))

2.1.1.2 Equilibrium equations

The equilibrium equations state that the sum of all forces acting on an object should be 0. For static problems, there is by definition no acceleration and the equations reduce to:

$$\sigma_{ji,j} + f_i = 0 \tag{2.4}$$

where f_i represent the volumetric forces and σ_{ij} is the Cauchy stress tensor of second order. The comma $,i$ is used to denote a partial derivative with respect to the i^{th} component ($\frac{\partial}{\partial x_i}$).

The balance of moments yields symmetry for this tensor:

$$\sigma_{ij} = \sigma_{ji} \tag{2.5}$$

2.1.1.3 Constitutive Equations

In the current context, the constitutive equations correspond to the relations which link stress and strain. The simplest model is *Hooke's law*, which assumes a linear relationship between the stress and the strains. It can be written as:

$$\sigma_{ij} = C_{ijkl}\ \epsilon_{kl} \tag{2.6}$$

C_{ijkl} is known as the material matrix. Its components are computed based on material properties such as Young's modulus and Poisson's ratio. The formulation changes greatly depending on the assumptions used in the material model.

2.1.1.4 Boundary Conditions

Two types of boundary conditions can be defined. In mathematics, the *Dirichlet* boundary conditions specify values of the solution on a certain part of the boundary. In the current context, it corresponds to displacements. Boundary conditions of *Neumann* type dictates the value of the derivative of the solution on a certain part of the boundary. In structural mechanics, this is equivalent to specifying a load vector on the boundary. A prescribed load vector $\mathbf{t}$ is related to the stress tensor σ and the normal vector $\mathbf{n}$:

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

2.1.1.5 Principle of Virtual Work

The principle of virtual work, also called principle of virtual displacements in some literature, states that for any displacement applied on the object its internal and external work remain balanced. By multiplying equation 2.4 with a virtual displacement $\delta\mathbf{u}$ and then integrating over the whole domain, one obtains:

$$\delta\Pi = \int_{\Omega} \delta u_i [\sigma_{ji,j} + f_i]\, d\Omega = 0 \tag{2.8}$$

which corresponds to the *strong* form, since the stress tensor has to be at least C^1 continuous. By integrating by parts and applying a variant of Green's theorem, the following equation is obtained:

$$\delta\Pi = \int_{\Omega} \delta\epsilon_{ij}\sigma_{ij}\, d\Omega - \int_{\Omega} \delta u_i f_i\, d\Omega - \int_{\Gamma_N} \delta u_i t_i\, d\Gamma_N = 0 \tag{2.9}$$

where Γ_N corresponds to the portion of the boundary where a Neumann boundary conditions are defined. The virtual strains ϵ_{ij} can be expressed in terms of the virtual displacements:

$$\delta\epsilon_{ij} = \frac{1}{2}(\delta u_{i,j} + \delta u_{j,i}) \tag{2.10}$$

The virtual displacements δu are defined such that they vanish at the sections of the boundary where a Dirichlet type boundary condition is applied. Γ_N corresponds to the portion of the boundary where a Neumann boundary conditions are defined. In equation 2.9, the requirements of continuity for the stress tensor are only C^0, hence it is called the *weak* form.

The first term of equation 2.9 corresponds to the work of internal stresses. The second and third one express the work done by external volumetric and traction loads, respectively.

The problem can then be written as:

$$a(\mathbf{u}, \delta\mathbf{u}) = \ell(\delta\mathbf{u}) \tag{2.11}$$

where $a(\cdot, \cdot)$ is the bilinear form of the internal work:

$$a(\mathbf{u}, \delta\mathbf{u}) = \int_\Omega \delta\epsilon_{ij}(\delta\mathbf{u})\sigma_{ij}(\mathbf{u})\, d\Omega \tag{2.12}$$

and $\ell(\cdot)$ is equivalent to:

$$\ell(\delta\mathbf{u}) = \int_\Omega \delta u_i f_i\, d\Omega + \int_{\Gamma_N} \delta u_i t_i\, d\Gamma_N \tag{2.13}$$

The problem can now be stated as: Find $\mathbf{u} \in K$ such that equation 2.11 is respected for every $\delta\mathbf{u} \in K$, where K is the space of the test functions defined as:

$$K = \{\delta\mathbf{u} \in H^1(\Omega) \mid \delta\mathbf{u} = 0 \text{ on } \Gamma_D\} \tag{2.14}$$

2.1.1.6 Discretization

For most applications, the analytical solution $\mathbf{u}$ to the equation 2.11 cannot be computed. Instead, the finite element method is applied, which subdivides the studied domain into individual elements. The behavior of these elements is well-known and by merging the reactions of these individual elements back together, one can obtain an approximation of the solution for the global problem. Naturally, the solution obtained depends a lot on the chosen discretization. Care has to be taken at this step in order to generate a mesh which represents the original geometry well while avoiding possible sources of errors such as singularities, see Zienkiewicz and Taylor (2005).

There are many different formulations for the finite elements, and research is still ongoing to find more efficient and special elements of specific applications. In the classical FEM, an element consists of a series of points, called *nodes*, at which the solution is evaluated. The quantities of interest are then reconstructed over the element domain using *shape functions*. For linear elements, each element has the same number of shape functions as its number of nodes. By definition, every shape function has a value of 1 at its associated node and a value of 0 at all others. For a linear one-dimensional element (2 nodes), this corresponds to, in parametric space:

$$\begin{aligned} N_1(\xi) &= \tfrac{1}{2}(1 - \xi) \\ N_2(\xi) &= \tfrac{1}{2}(1 + \xi) \end{aligned} \tag{2.15}$$

where ξ is the parametric coordinate of the element. Using this definition, the quantities of interest can be spanned across the studied domain to obtain a finite element approximation:

$$\mathbf{p} = \sum_{i=1}^{n} N_i p_i \tag{2.16}$$

where n is the number of shape functions used and $\mathbf{p}$ can represent any physical quantity of interest such as stresses or displacements.

2.1.2 High Order Finite Elements

Over the years, there has been an important interest in research for alternate formulations of the FEM. This has led to the development of the high-order finite element method, or p-FEM. This version uses a new type of elements which use of shape functions based on high-order polynomials. In contrast to the linear elements used in the classical FEM, p-fem has shown an exponential convergence rate for smooth problems, a better representation of the geometry and reduced locking problems (Szabó et al. (2004)). Moreover, for many problems, it has proved to generate more accurate results for a lower number of degrees of freedom.

2.1.2.1 High Order Shape Functions

There are two main possibilities for the polynomials used in the shape functions of high-order elements. The classical one, based on Lagrange polynomials and the so-called hierarchic shape functions, based on integrated Legendre polynomials Düster (2008). Figure 2.1 gives an overview of these shape functions in one dimension for different polynomial orders. The main difference between the two is that the lower-order shape functions are kept in the higher-order basis.

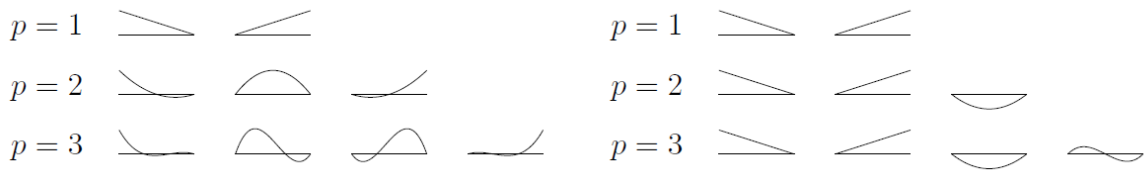


Figure 2.1: One-dimensional standard (left) and hierarchic (right) shape functions for p=1,2,3 (Franke (2011))

2.1.2.2 Extension to 2D and 3D

The problems studied in the course of this work are in two and three dimensions. To construct the necessary 2D and 3D elements, additional shape functions have to be defined. They are obtained by taking the tensor product of the one-dimensional shape functions:

$$N_{i,j}^{2D}(r,s) = N_i^{1D}(r) \otimes N_j^{1D}(s) \tag{2.17}$$

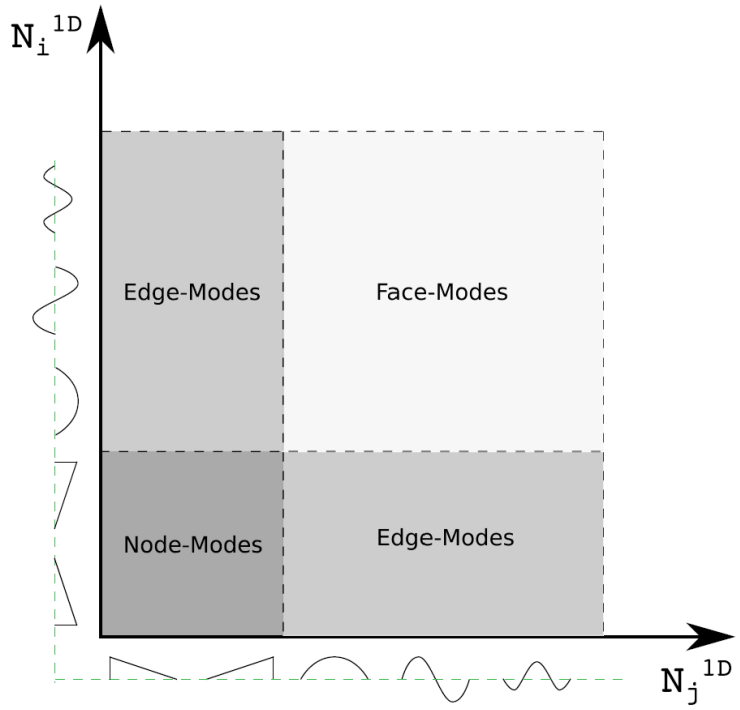


Figure 2.2: Tensor product for two-dimensional shape functions (Zander et al. (2014))

Similarly, for the three-dimensional case:

$$N^{3D}_{i,j,k}(r,s,t) = N^{2D}_{i,j}(r,s) \otimes N^{1D}_{k}(t) \tag{2.18}$$

This tensor product can be visualized in figure 2.2. This leads to new shape functions which are not necessarily associated with a node, but also with an edge or with the interior of the domain. *Nodal* shape functions refer to the ones that take the value 1 at a specific node and 0 at all the other ones, like in the classical linear case. *Edge* shape functions, analogously, are non-zero along a certain edge and zero on all others. The third type is called *face* or *bubble* shape functions. These are zero at all edges and nodes but non-zero inside the domain.

2.2 The Finite Cell Method

The finite cell method can be classified as a fictitious domain or an embedded domain method (Vos et al. (2008)). It is a modification of the classical finite element method. It combines the advantages of the p-FEM along with a special integration technique. This allows the computation of complex structures without the necessity to generate a mesh. This method was developed at the Chair for Computation in Engineering of the Technische Universität München (Parvizian et al. (2007), Düster et al. (2008)).

This section gives an overview of the finite cell method. A complete, comprehensive monograph on the FCM can be found in Schillinger and Ruess (2014). This section gives an overview of the main particularities of the finite cell method. It is divided into two parts. The first part covers some of the main particularities of the FCM, such as the fictitious domain approach, the spatial integration technique, the weak treatment of boundary conditions

and the applicability to large deformations. Afterwards, some remarks more closely related to the present work are made, which include the algorithm used to recover the boundary and the geometrical update that is performed.

2.2.1 Fictitious Domain Approach

The main goal of the FCM is to eliminate the need for a suited mesh for the analysis of complex structure. This is done by embedding the studied body in a regular Cartesian grid of high-order elements, as in figure 2.3. The problem domain is now composed of two individual domains:

$$\Omega_U = \Omega_{phy} \cup \Omega_{fic} \tag{2.19}$$

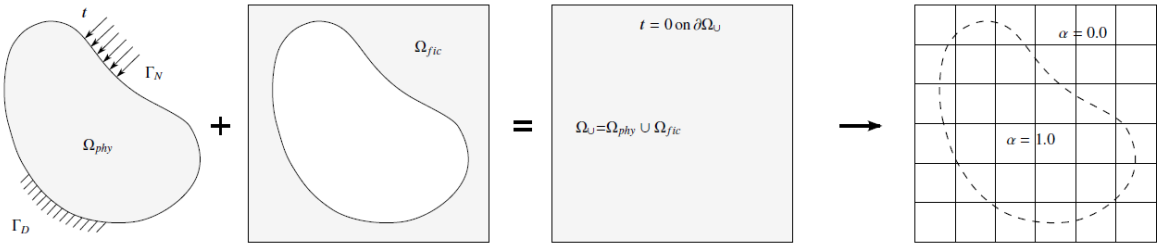


Figure 2.3: Finite cell method: Embedded object in high-order finite elements (Schillinger et al. (2012))

An indicator function α is then defined:

$$\alpha(x) = \begin{cases} 1 & \forall\ x \in \Omega_{phy} \\ 0 & \forall\ x \notin \Omega_{phy} \end{cases} \tag{2.20}$$

To prevent bad conditioning of the stiffness matrix, a value of approximatively 10^{-10} is used in numerical implementations. This parameter is then combined with the bilinear form in equation 2.11 to cancel out the non-physical contributions of the fictitious domain to the weak form.

2.2.2 Spatial Integration

In the FCM, the geometrical boundaries are no longer defined by the finite element mesh, but by the Gauss points during the integration. However, a key requirement of the Gaussian quadrature is the smoothness of the integrand. Unfortunately, the parameter α introduced in equation 2.20 introduces some discontinuities in the integrands. To circumvent this problem, the finite cells are successively cut, as shown in figure 2.4, leading to a space-tree integration mesh. Once the desired depth is obtained, a simple inside-out test is performed on the Gauss points to integrate the desired quantities.

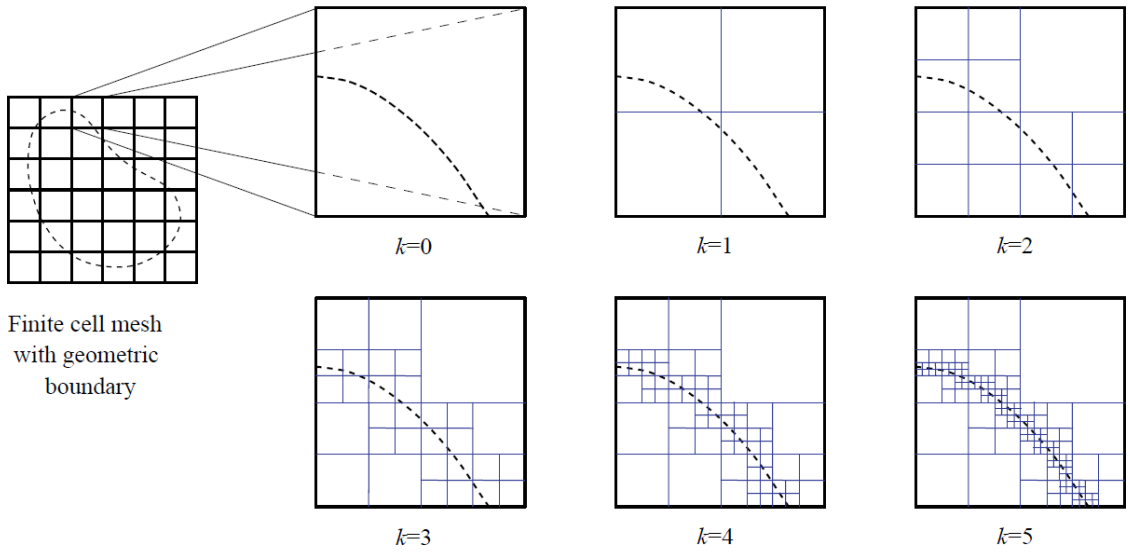


Figure 2.4: Integration mesh for the finite cell method (Schillinger and Ruess (2014))

2.2.3 Weak Boundary Conditions

The enforcement of Neumann boundary conditions poses no difficulties in a FCM context, since they are taken into consideration by an additional term in the weak form of the problem, see equation 2.13. The Dirichlet boundary conditions, however, bring forth new challenges resulting from the dissociation of the finite element mesh and the geometry representation. This means that a new way of imposing Dirichlet boundary conditions has to be defined without using a conforming mesh.

This is done through an extension of the weak form of equation 2.12, hence this technique is referred to as *weak enforcement* of boundary conditions. Many methods have been proposed in the literature based on the penalty method or Lagrange multipliers or the formulation introduced by Nitsche (Nitsche (1971)). Due to its simplicity, the penalty method is the most commonly used method for FCM.

The following potential is defined:

$$\Pi^P(\mathbf{u}) = \frac{1}{2}\beta \int_{\Gamma_D} (\mathbf{u} - \hat{\mathbf{u}})^2 \, d\Gamma_D \tag{2.21}$$

where β is a penalty parameter and $\hat{\mathbf{u}}$ is the prescribed displacement. The extension of the weak form is done through the addition of the variational form of equation 2.21, which can be expressed as:

$$\delta\Pi^P(\mathbf{u}) = \beta \int_{\Gamma_D} (\mathbf{u} - \hat{\mathbf{u}})\delta\mathbf{u} \, d\Gamma_D \tag{2.22}$$

The most important drawback of this method is the effect of the penalty parameter β. When a high value is chosen, it can deteriorate the condition number of the resulting stiffness matrix drastically and lead to convergence problems in the solution process.

2.2.4 Finite Cell Method for Large Deformations

There are many sources of non-linearity for static problems. Problems are said to be non-linear when the response (output) is not directly proportional with respect to the input. For example, in the case of a static problem, the displacements and internal stresses should be doubled when the applied forces are doubled. There are 4 main sources of nonlinearity for structural problems: material, geometry, force boundary conditions or displacement boundary conditions, see Felippa (2015). For problems involving large deformations, the assumptions made in section 2.1.1 are no longer valid.

For this purpose, a non-linear material model has to be used. Throughout the present work, bodies subject to large deformations are modeled using a hyperelastic, compressible neo-Hookean solids. Their strain energy is expressed as:

$$W_{NH} = \frac{G}{2}(\text{tr } \hat{\mathbf{b}} - 3) + \frac{\kappa}{2}(J-1)^2 \tag{2.23}$$

where G is the shear modulus, κ is the initial bulk modulus, J is the determinant of the gradient of deformation and $\hat{\mathbf{b}}$ is the deviatoric part of the left Cauchy-Green tensor, see Bonet and Wood (2008). The stress-strain curve associated with this model is initially close to linear but experiences a plateau starting at a certain value of deformation. However, the model assumes perfect elasticity and thus no dissipation of energy is accounted for.

The penalization parameter presented in equation 2.20 introduces some important discontinuities in the strains and displacements. For very low penalization values $\alpha \approx 10^{-10}$, the deformation gradient falls below 0, which results in the deformation mapping having non-unique solutions and the solution process to terminate (Schillinger and Ruess (2014)). Therefore, the penalization factor has to be increased, which then causes non-negligible modeling errors as the influence of Ω_{fic} is more important. This effect leads to severe oscillations which can cause overlapping of the elements or, at the very least, a harsh deterioration of the convergence rate.

To counteract this effect, a simple trick is used to minimize the contributions of the fictitious domain. Since the values of the solution in Ω_{fic} are of no practical interest, it is possible to reset them at every iteration of the Newton-Raphson solver used. Hence, the deformation mapping is reset to the initial configuration at every step, deleting the deformation history for the fictitious domain. Since the deformation of the fictitious domain starts from 0 at every load step, it is assumed that linear elastic deformations take place at every step. Therefore, from a computational efficiency point of view, it is much more advantageous to use a linear elastic model for the fictional domain part. This leads to a smooth extension of the solution in the fictitious domain and allows the use of low values for the penalization parameter. The resulting formulation is inconsistent and violates some key principles of continuum mechanics such as the penetration of material. However, it does not affect Ω_{phy}, which is the domain of interest. A more detailed analysis of this technique along with numerical examples can be found in Schillinger and Ruess (2014).

2.2.5 Marching Squares and Marching Cubes

Since FCM is an embedded domain method, the boundary of the studied body is not obtained directly from the discretization. For the enforcement of constraints such as contact, a parametric surface is often required and thus needs to be reconstructed. As it was pointed out by Kim et al. (2007), it is advantageous to use a discretization for the boundaries that is independent of the embedding mesh. This prevents some possible complications such as a bad discretization for the Lagrange multiplier field which lead to oscillations and can make the enforcement of constraints difficult. It also facilitates the implementation of error estimators and adaptivity methods, since it is not necessary to ensure that the mesh still conforms to the geometry after every refinement step.

The technique used for the reconstruction of the surface is known as Marching Cubes, or Marching Squares for two-dimensional applications. It was first introduced in Lorensen and Cline (1987). It uses a divide-and-conquer approach to reconstruct the surfaces. The *embedding* domain is first split into a series of cubes, depending on the desired accuracy of the reconstruction by the user. Each cube is tested to see if the *embedded* domain passes through it.

In order to reconstruct the surface, it is necessary to figure out in which way the surface intersects with every given cube. They are a total of 256 possibilities. These can be summarized by 15 different cases, the rest can be obtained by symmetry. The original illustration of the 15 cases from Lorensen and Cline (1987) can be seen in figure 2.5. The intersection type is determined with the help of the cube vertices. It assigns a certain value to every vertex, depending on whether or not this vertex is inside the embedded body. The values of all vertices are then associated with one of the 256 cases and an inexact line search technique is used to find the intersection between the surface and the different edges. The necessary triangles are created and the algorithm then moves on to the next cube.

The Marching Squares algorithm proceeds in a similar way. It separates the two-dimensional field using a grid. Each square is then treated independently and the intersection of the surface with the edges is obtained again via inexact line search. The lookup table has a total of 16 entries.

Once all the cubes or squares have been processed, the complete reconstructed surface can be obtained. This leads to a coherent tessellation of the boundary of the embedded domain which can afterwards be used for the set up and the definition of the contact problem. Moreover, the resulting geometrical entities conform with the finite cells, which allows numerical integration to be carried out along these entities without integration error.

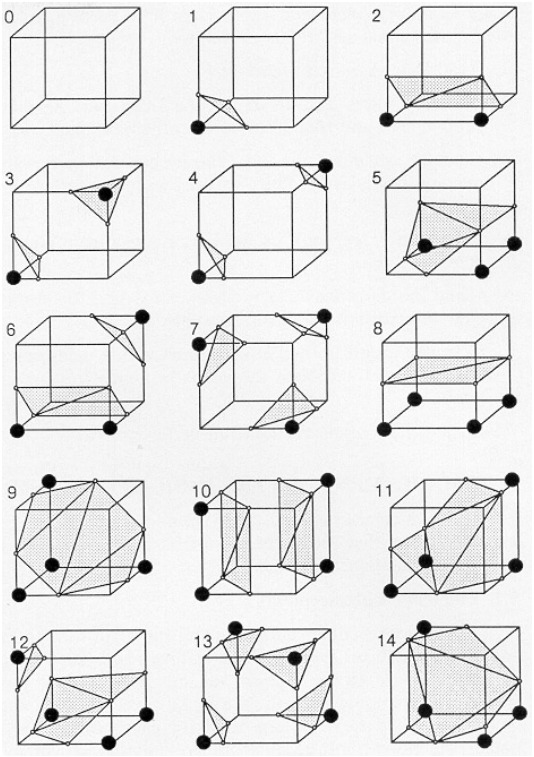


Figure 2.5: Marching Cubes lookup tables: 15 cases. Taken from Lorensen and Cline (1987). Used with permission of the Association for Computing Machinery (ACM).

2.2.6 Finite Cell Geometrical Update

The finite cell method dissociates the concept of element and geometry approximation inherent to the classical finite element method. Moreover, the total Lagrangian description used means that an update of the geometrical boundaries is required in order to obtain a current version of the geometry.

The geometrical boundary of the object is reconstructed using the algorithm presented in section 2.2.5. The result is a set of linear geometrical entities which are called *segments*. To obtain the current version of a segment, it is necessary to deform the initial one using the solution of its embedding element. In the case of linear segments, such as the ones obtained from the marching cubes or squares algorithm, the element displacement values only have to be applied to the cornering vertices. This leads to a very computationally inexpensive method combines some of the advantages of high-order finite elements along with the simplicity of linear boundaries. However, this simple approach comes at a certain expense: the true shape of the boundary inside this segment is, in most practical cases, not linear and therefore an error is made. Moreover, since the solution is spanned by high-order shape functions in the FCM context, linear segments would not stay linear in case of general deformation. A very fine discretization with the marching squares and cubes is usually employed to minimize the effects of this error.

Chapter 3

Introduction to Computational Contact Mechanics

This chapter introduces the generalities related to the simulation of problems involving contact. It follows the works of Wriggers (2006) and Yastrebov (2011). The first section defines the general problem studied in the field of computational contact mechanics. Afterwards, the conditions used to treat contact problems are presented, followed by an overview of the main approaches available to enforce contact constraints. Finally, the different methods used for the discretization of contact problems are introduced.

3.1 Problem Definition

The most general case for contact is illustrated in figure 3.1. It consists of two bodies, Ω_1 and Ω_2, which are in contact along their common boundary Γ_c. The sections of the boundaries where a prescribed force is applied are denoted Γ_N while Γ_D is used for prescribed displacements. It leads to a boundary value problem which can be described using the following equations:

$$\begin{cases} -\Delta u = f & \text{on } \Omega_{1,2} \\ u = u_0 & \text{on } \Gamma_D \\ \mathbf{n} \cdot \nabla u = t & \text{on } \Gamma_N \\ ? & \text{on } \Gamma_c \end{cases} \tag{3.1}$$

The most common approach to the analysis of contact problems is to consider it as a constrained optimization problem, see Wriggers (2006). The objective function is taken as the potential energy of the system which needs to be minimized:

$$\begin{cases} \min & \Pi(\mathbf{u}) & = & \Pi_{int}(\mathbf{u}) + \Pi_{body}(\mathbf{u}) + \Pi_{ext}(\mathbf{u}) + \Pi_C(\mathbf{u}) \\ \text{such that} & g(\mathbf{u}) & \geq & 0 \end{cases} \tag{3.2}$$

where the terms on the right side of the first equation correspond to the internal energy, the applied body (or volumetric) loads, the applied boundary loads and the contact potential energy, respectively. This corresponds to the equations introduced in chapter 2, with the addition of the contact term and the constraint. The constraint $g(\mathbf{u}) \geq 0$ represents the impenetrability condition, where $g(\mathbf{u})$ is the gap between the two bodies. In order to minimize

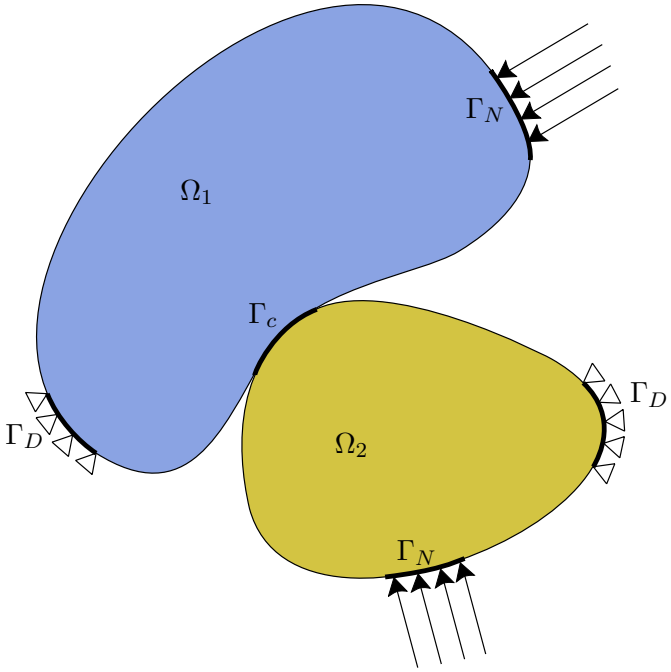


Figure 3.1: General contact problem between two deformable bodies

the potential energy defined in equation 3.2, the principle of virtual work can be applied (Bonet and Wood (2008)) which leads to the following variational form:

$$\delta\Pi(\mathbf{u}) = \delta\Pi_{int}(\mathbf{u}) + \delta\Pi_{body}(\mathbf{u}) + \delta\Pi_{ext}(\mathbf{u}) + \delta\Pi_C(\mathbf{u}) \tag{3.3}$$

For the current problem, the potential energy in equation 3.2 can be formulated using the equations 3.1:

$$\int_\Omega \nabla\mathbf{u} \cdot \nabla\delta\mathbf{u}\ d\Omega = \int_\Omega f\delta\mathbf{u}\ d\Omega + \int_{\Gamma_N} t\delta\mathbf{u}\ ds + \int_{\Gamma_C} C\ d\Gamma_c \tag{3.4}$$

where $\mathbf{u} \in K$ is sought and $\delta\mathbf{u}$ are the test functions. This formulation is equivalent to the one presented in section 2.1.1.5, only with an additional contact term.

The penalty functional in equation 3.2 is taken as:

$$\Pi_C(\mathbf{u}) = \frac{1}{2}\int_{\Gamma_C} \epsilon_N g(\mathbf{u})^2 \tag{3.5}$$

where ϵ_N is the penalty parameter and g is the gap or penetration between the two bodies. By taking the variation of equation 3.5, an expression for the last term of equation 3.4 can be obtained:

$$C = \epsilon_N g(\mathbf{u})\delta g(\mathbf{u}) \tag{3.6}$$

It is now possible to define the normal force applied to both bodies in contact:

$$N = \epsilon_N g \tag{3.7}$$

A physical analogy can be made with a spring-mass system, where a spring of stiffness ϵ_N

creates a force N on the mass when it penetrates the rigid surface by a value of g, see figure 3.2.

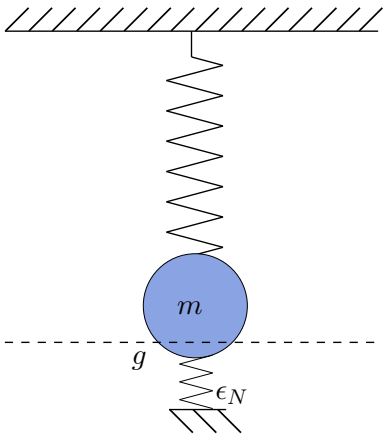


Figure 3.2: Analogy of the penalty regularization for contact to a spring-mass system

The penalty term ϵ_N is a decisive factor for the numerical simulation of contact. If this term is too small, the penetration is not penalized enough and it results in large intersections between the contacting bodies. If on the contrary this term is too large, the convergence of the solution is difficult due to the ill-conditioning of the tangent matrix. The correct choice of this parameter is therefore crucial. Although not discussed here, it should be noted that there are also automatic and adaptive techniques which can be used to set this parameter during the simulation (Persson (2000)).

3.2 Contact Conditions

3.2.1 Normal Contact Conditions

One of the main factors in the simulation of contact is the normal force acting on both bodies. Instead of defining constitutive equations in the contact interface, some mathematical conditions are used to compute this normal force, see Wriggers (2006). This force is non-zero only when the gap is 0 (see equation 3.2) and it should be equal on both sides of the contact interface. This can be formulated using the Hertz-Signorini-Moreau conditions:

1. If a positive gap exists between the two bodies, the normal reaction force should be 0.

$$g > 0 \rightarrow N = 0 \tag{3.8}$$

2. If the two bodies touch, or the distance between them is 0, the normal reaction force should be greater than 0.

$$g = 0 \rightarrow N \geq 0 \tag{3.9}$$

3. These two statements can be resumed by the following third complimentary condition:

$$gN = 0 \tag{3.10}$$

These conditions are also known as the Karush-Kuhn-Tucker conditions in the field of optimization. They can be visualized by plotting the normal reaction force with respect to the gap, as in figure 3.3.

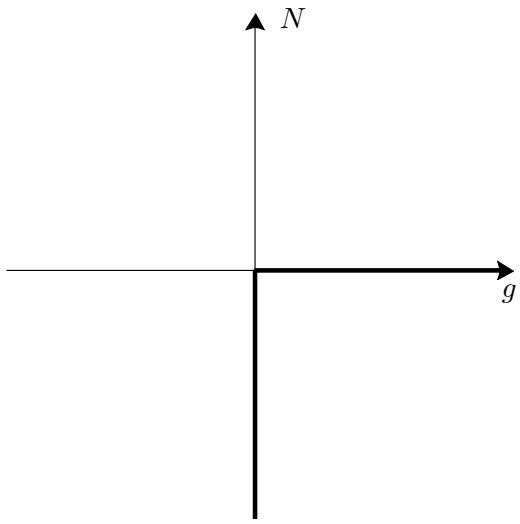


Figure 3.3: Normal contact force versus normal gap

3.2.2 Tangential Contact Conditions

For cases where friction is considered, some additional conditions are required. The treatment of the tangential tractions T_i (components of the reaction force which are tangent to the interface) differs depending if the system is in a stick or slip state. The mathematical modeling of friction can be very complex. One of the most commonly used and simplest formulation is Coulomb's law. It states that two bodies are sticking together until a certain value is reached for the tangential force. It can be summarized by the following equations:

1. If the two bodies are not in contact, there is no normal or tangential forces:

$$g > 0 \rightarrow N = T = 0 \tag{3.11}$$

2. If the two bodies are in contact and their relative tangential velocity is 0, sticking is observed:

$$v_t = 0 \rightarrow \text{Stick} \rightarrow |T| - \mu|N| < 0 \tag{3.12}$$

3. If the two bodies are in contact and their relative tangential velocity differs from 0, sliding is observed:

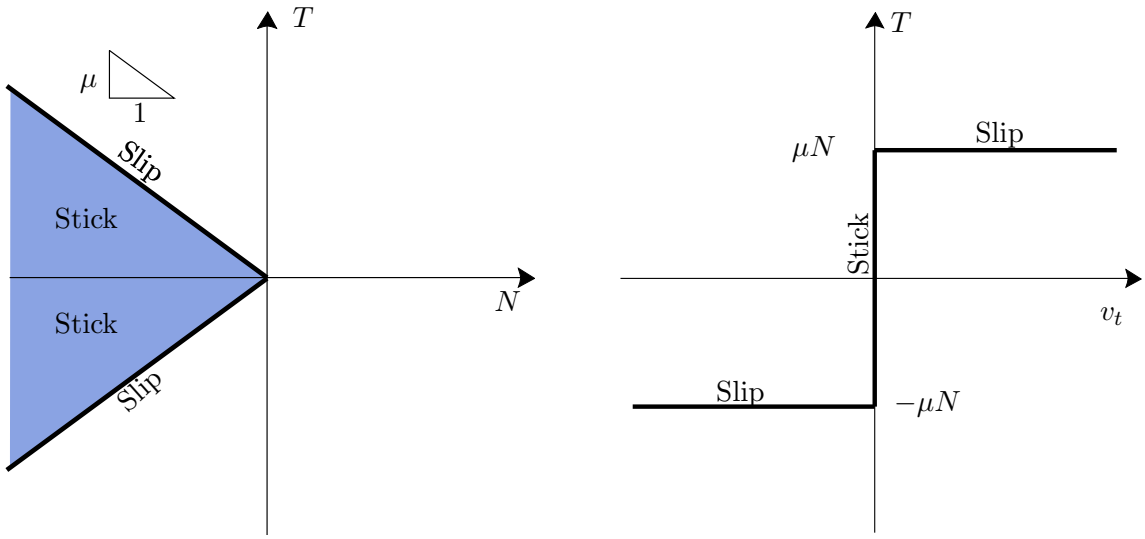
$$|v_t| > 0 \rightarrow \text{Slide} \rightarrow |T| - \mu|N| = 0 \tag{3.13}$$

4. These two statements can be resumed by the following third complimentary condition:

$$|v_t|(|T| - \mu|N|) = 0 \tag{3.14}$$

In these equations, v_t corresponds to the tangential velocity and μ is a constant called the coefficient of friction, which depends mainly on the materials and properties of the surfaces in

contact. These conditions can be visualized in figure 3.4, which illustrate the two-dimensional case. In three dimensions, the Coulomb slip surface takes the shape of a cone for the normal tractions versus tangential tractions plot and of a cylinder for the norm of the tangential velocity versus the tangential tractions plot. The Coulomb law for friction, although simplistic, is valid for most problems of practical relevance. In reality, the coefficient of friction μ sometimes differs between the stick and slip states. This value also varies slightly depending on the relative velocity during the slip state. Some models have been proposed to take these effects into account but such models have been deemed unnecessarily complex for the purpose of the present work (Persson (2000)).



(a) Tangential force versus normal force **(b)** Tangential force versus tangential velocity

Figure 3.4: Illustration of the frictional contact conditions in 2D

3.3 Contact Constraint Enforcement

There is a wide range of methods to apply contact boundary conditions between two bodies, although very few of them are currently implemented in commercial software. These methods originate for the most part from the optimization theory. The most common methods are the following: Lagrange multiplier, penalty, augmented Lagrangian, barrier, and Nitsche's method.

Lagrange multiplier methods add additional degrees of freedom. A new functional is defined, the *Lagrangian*. A search for the saddle point of this new functional is then performed. The biggest advantage of this method is that it leads to an exact solution. In the computational contact context, this implies that there will be no penetration between the two bodies. However, this advantage comes at a cost: the aforementioned additional degrees of freedom and the saddle point structure, which restricts the choice of applicable solvers.

The augmented Lagrangian method was developed in an attempt to improve the Lagrange multiplier method. Just as the latter, it leads to the exact solution. It is a combination of the Lagrange multiplier method and the penalty method. The improvement is mainly in terms

of smoothness, as it leads to a C^1 differentiable saddle point functional. Methods of this type are often combined with Uzawa algorithms, which allow to solve the weak form in an inner loop while the Lagrange parameters are held constant in an outer loop during the iteration, see Wriggers (2006).

Barrier methods are a special branch of methods which differs from the ones aforementioned as all constraints are permanently active. In the current context, this signifies that there is no contact search step required to determine the contact areas as is the case for the other methods. The initially constrained problem is then changed into a series of unconstrained ones. However, the tangent matrices can easily become ill-conditioned. A barrier-type method has been implemented successfully in a high-order finite element context by Bog et al. (2015). From a physical point of view, the physical domain is embedded in a larger fictitious domain with a special material formulation to prevent the penetration of surfaces. This material formulation starts stiffening when it is heavily compressed. This technique has provided very promising results. However, the contacting surfaces should coincide with the element boundaries in order to prevent oscillations in the solution, which is conflicting with the general FCM idea.

For Nitsche's method, the stress vector in the contact interface is computed as a mean value of the two contacting bodies. This leads to an exact enforcement of the constraints without the need for additional degrees of freedom. The Nitsche method can become quite complex in case of non-linearities and in such cases it is advisable to use a Lagrangian multiplier or penalty formulation.

For the penalty method, which is the method of choice for the present work, a new functional is defined as the sum of the functional to be minimized and a penalty term. This penalty term is based on a penalty parameter ϵ and the gap (or penetration) between the two bodies. Penalty methods transform constrained problems into unconstrained ones by penalizing infeasible solutions. They are robust and usually simpler than other methods. The solution is approximated but tends to the mathematically exact solution as the penalty term tends to infinity, see Luenberger (1984). The main advantages of this method are its simple implementation and the fact that, as opposed to the Lagrangian method, it does not add extra degrees of freedom to the problem. Its biggest flaw would be the inexact solution as only an approximation of the exact solution is obtained.

The techniques listed here is far from complete and constitute just the tip of the iceberg that is the field of computational contact mechanics. Wriggers (2006) provides a more complete list with exhaustive explanations.

3.4 Discretization Methods for Contact Problems

An important part of every contact algorithm is the discretization of the contact boundary. In a large deformation context, a body might come into contact with other bodies or even itself, leading to self-contact. Furthermore, there exists the possibility of relative sliding between two meshes, which implies that a certain slave node could move from one master element to another, for example. This must be taken into account in the chosen approach in such a way that the update of contact pairs and the repartition of reaction forces is done coherently, especially in the case of large deformation.

There are many different approaches used for the finite element discretization of the contact equations. For the purpose of the present work, a decisive factor for the choice of the discretization method is the possibility to use a high-order polynomial for the interpolation of both slave and master elements. It is also of particular importance to choose an approach which fulfills the *Babuška-Brezzi* conditions (also known as *BB* or *inf-sup* conditions). These conditions are necessary and sufficient to ensure a stable discretization scheme.

One of the simplest and most used methods is known as the "Node-To-Node" (NTN) approach. This approach considers the penetration between a node on the slave surface and a node on the master surface. Conforming meshes are required for the correct application of this method due to the necessity of having aligned nodes. This approach cannot support large tangential deformations (sliding of a surface along another). Another problem arises when it comes to defining the normal vectors at the node pairs. Since this method requires a linear discretization of both master and slave bodies, the C^1 continuity is not respected at the element boundaries.

The "Node-To-Segment" (NTS) approach differs from the previous one by the fact that the point on the master surface is now obtained from the projection of a slave node. This is appropriate for large deformation contact applications, however, as the slave *nodes* are used in this process, it is mostly used for linear discretizations of the slave surface, although it has also been extended to isogeometric analysis, see Matzen et al. (2013). Due to its robustness and versatility, this method is used extensively in commercial finite element software. Using this approach, one must be careful to choose the coarser mesh as the master surface since excessive interpenetration might occur otherwise. This is due to the fact that, if the coarser mesh is chosen to be the slave, the master nodes would be allowed to enter the slave surface without being detected.

The "Segment-To-Analytical-Surface" (STAS) method is briefly mentioned in Konyukhov and Izi (2015). It is used in applications where one of the interacting bodies can be approximated as a rigid surface. The deformable body is chosen to be the slave surface and the master surface geometry (or curve in 2D) is described analytically. The contact is then checked using the penetration of the discretized slave body through the master rigid surface. Although very promising for problems involving complex geometries, the current work focuses on contact between two deformable bodies and therefore cannot make use of this formulation.

Other methods make use of an artificially created contact interface. This allows the application of contact constraints to non-matching meshes. The discretization with *contact segments* (Wriggers (2006)) uses the projection of master and slave nodes onto the opposing surface to create an intermediate, C^1 continuous contact line based on Hermite polynomials.

The *Nitsche* and *mortar* methods use again a common reference interface which is obtained either through interpolation or using the surface of one of the contacting bodies. The idea of the Nitsche method is to compute a stress vector for the reference interface using an average value of the stresses in both contacting bodies. For the classical mortar method, on the other hand, Lagrange multipliers and the gap function are defined on the reference interface and are then interpolated. The penalty method can also be applied to the same formulation. This is referred to as the "Segment-To-Segment" or "Surface-To-Surface" (STS) method in Konyukhov and Izi (2015). This formulation was chosen for the present application due to its fulfillment of the BB conditions and its compatibility with arbitrary discretization for the surfaces, such as high-order polynomials or even isogeometric analysis applications. It does

not correspond, however, to the classical definition of the mortar method. The constraints are enforced in a weak sense using a penalty factor instead of the Lagrange multipliers. Since the integration is done on the slave body, integration points are spread across the slave segment and are then projected on the master surface (see chapter 5). The contributions to the virtual work are therefore included in a point wise manner, depending on the penetration of each individual Gauss point on the slave segment. The contact area is hence approximated using the slave integration points. This means that a higher number of evaluation points (based on a Lobatto or Gaussian quadrature) would most likely produce a sharper definition of the contact boundary.

Chapter 4

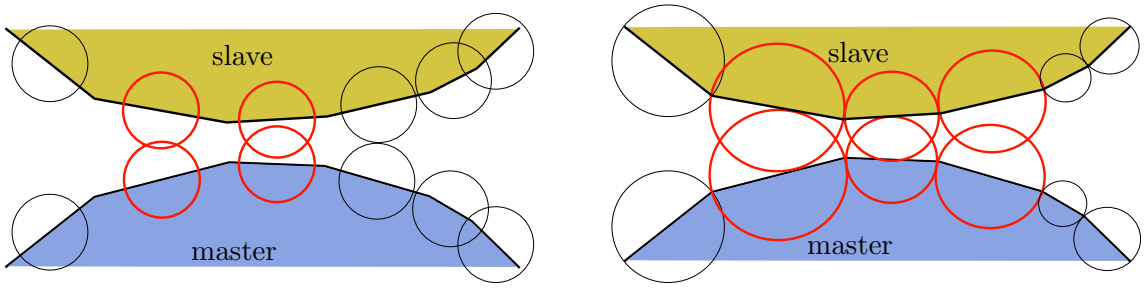
Global Search Algorithms

Global search algorithms are first used on the studied problem in order to simplify the analysis as contact mathematical models can be rather expensive to compute. The main goal of global search routines is therefore to save computation time.

Ideally, the contact mathematical model would be run only on components that are interfering when they should not be. Global search algorithms can be used in an attempt to narrow down the number of components on which the contact model needs to be applied. The easiest way to save a lot of resources is if there is *a priori* knowledge of the evolution of the problem or if there exists a coherence in the configuration between load or time steps. This allows the use of assumptions for future steps. Methods which take advantage of these particularities are called *non-exhaustive*. When large, unpredictable deformations are present, the algorithms need to be run at every load or time step of the analysis over all the objects. In this case, *exhaustive* methods are run based only the information available at the current load or time step. This is done in an exclusively geometrical manner, i.e. only the position of the objects in space are considered. Exhaustive algorithms are therefore more robust but also slower in general, as is discussed in Williams and O'Connor (1995).

For the present work, exhaustive methods are required due to the presence of large deformations. The reconstruction of the boundary is done via a marching cubes algorithm (see section 2.2.5) and leads to a polygonal mesh of linear segments which approximate the surface of the geometry. The main idea is to run a simple and efficient algorithm over these individual linear segments and produce a set of *potentially* interfering segments, which will be referred to as *contact pairs*. These contact pairs contain, in the context of the finite cell method, a geometrical entity and its associated high-order element for both master and slave objects. These pairings are then checked at a later step to see if there actually is interference between them. This is done via a closest-point projection technique and is discussed in chapter 5.

The difficulty in defining a single or perfect global search formulation resides in the high diversity of problem types. Moreover, special cases such as problems including self-contact or flow of a high number of particles require special considerations. Many formulations have been proposed (Wriggers (2006)) to address these difficulties, each with its own advantages and disadvantages. For the current work, two algorithms have been implemented and are presented here: pinball contact search and spatial sorting contact search.



(a) Using a fixed chosen value for the radius **(b)** Using the size of the segment as diameter

Figure 4.1: 2D illustration of the pinball global search

4.1 Pinball Global Search

The pinball global search is probably the simplest formulation and consequently the easiest to implement. The idea is to create a circle (or a sphere for three dimensional problems) centered on every segment midpoint and check for contact between those spheres.

At every load or time step, the current position of every segment of the surfaces is determined. The center of gravity of the segments (which corresponds to the midpoint in 2D) is then computed. The distance between midpoints is calculated and compared to the pinball radius threshold as in the following equation:

$$|(\rho_{CoG} - r_{CoG})| \leq D_{pinball} \tag{4.1}$$

where ρ_{CoG} and r_{CoG} refer to the master and slave midpoints, respectively. Satisfying this inequality means that the pinballs are interfering as is shown in red in figure 4.1. The master-slave pair are saved in a list of contact pairs. The pinball radius $R_{pinball} = D_{pinball}/2$ can be computed in two different ways. A custom radius can be specified and is thereafter be used on every segment (see figure 4.1a)). If no value is provided, a default value is computed based on a simple minimum bounding circle or bounding sphere algorithm which corresponds to the smallest radius ensuring containment of all the nodes of the segment (see figure 4.1b)). In this case the value is simply obtained from equation 4.2. One can note that, depending on the chosen definition of $R_{pinball}$, the list of contact pairs can vary.

$$D_{pinball} = R_{ma,bounding} + R_{sl,bounding} \tag{4.2}$$

This algorithm, although robust and simple, has one major disadvantage. At every load or time step, equation 4.1 has to be evaluated for every slave segment with all the segments of the master boundary. In case of self-contact, every segment has to be checked with all the other segments from the object's boundary. This means that the order of checks required is $N_{master} * N_{slave}$ $(O(N^2))$, where N represents the number of segments on the boundary. For very fine discretizations, this procedure can be quite slow and inefficient, since technically not all segments should have to be checked, but only those in a certain range around the studied segment. This is what motivated the derivation of other formulations, such as the sorting global search, which is the topic of the next section.

4.2 Spatial Sorting Global Search

4.2.1 Spatial Sorting Technique

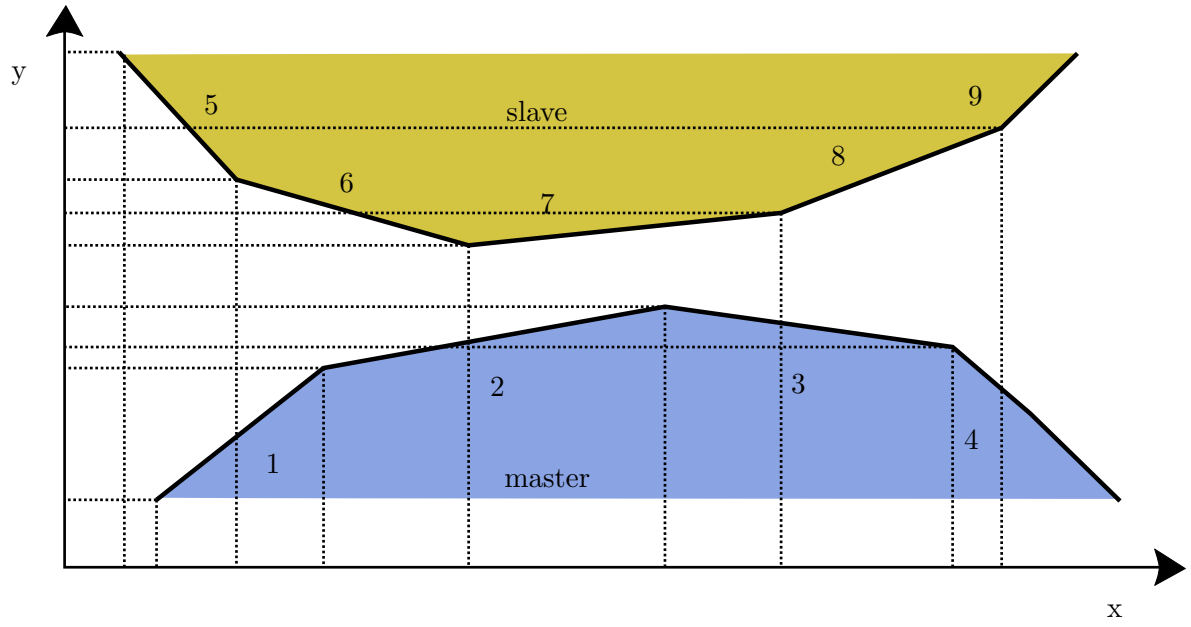
The spatial sorting global search was implemented to lower the computational costs of the classical pinball contact search algorithm. It is inspired by the works of Williams and O'Connor (1995) and O'Connor (1996). The authors describe therein the heapsort spatial sorting technique, which is highly efficient and versatile since it can be applied to an arbitrary number of objects of random shapes and sizes. The algorithm implemented in the course of this work is an adapted version of the one described in these articles. It is used as a global search tool in the cases of self-contact or elastic-elastic contact between two bodies. The general idea remains the same, however: to find the minimum and maximum coordinates of every object along the principal axes of the global Cartesian space and to sort them in an efficient way.

The first step is to find the minimum and maximum ordinates of each segment in the current configuration. For the purpose of the current work, the algorithm is used to determine the sections of boundaries which are in contact between two elastic bodies. Those boundaries consist of linear segments in the case of a 2 dimensional problem or surface elements (quadrilaterals or triangles) for 3 dimensional problems. Note that in its existing formulation, this min/max technique cannot be applied to these kinds of objects. If one of the segments happens to be parallel to one of the principal axes (or principal planes in 3D) it creates ambiguities for the sorting algorithm. This ambiguity comes from the fact that if a segment is vertical (parallel to the y axis) in the x-y plane, the algorithm has trouble finding the minimal and maximal values in the x direction as well as sorting it with respect its neighbors. Thus a different technique is employed, one that one that is related to the pinball global search. For every segment, the midpoint is computed and an imaginary pinball is centered at this position. This removes the necessity to access all nodes and compare them to find the minimum and maximum value, having instead only to map the origin in the local space to the global space to obtain the midpoint. Figure 4.2 shows the difference between the min/max approach and the so-called pinball style min/max approach. Only the minimum lines were drawn for the sake of clarity. As is the case for the pinball global search algorithm, it is possible to use a constant pinball radius or a default value based on the minimum bounding circle (or sphere). Once the midpoint position is known in the global space, the computation of minimum and maximum values is straightforward.

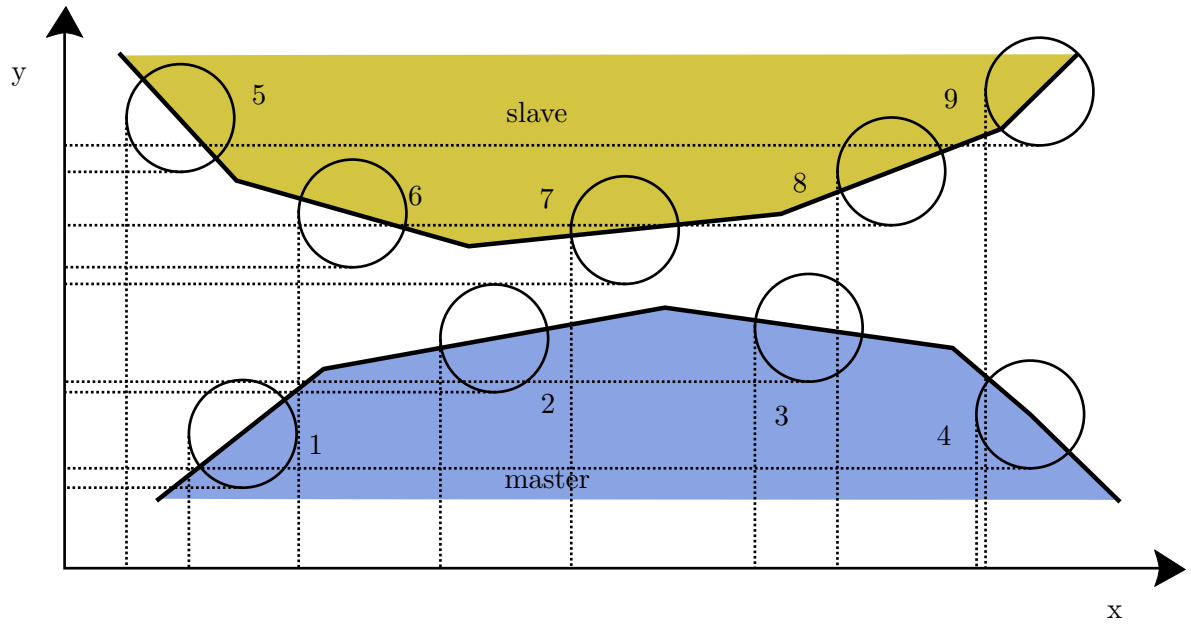
$$X_{MAX} = X_{midpoint} + R_{pinball} \tag{4.3}$$

$$X_{MIN} = X_{midpoint} - R_{pinball} \tag{4.4}$$

Once the min/max values for every segment have been obtained, the next step is to sort the objects. All the segments (from both slave and master boundaries) are merged together and sorted. The algorithm described by Williams and O'Connor (1995) uses a special heapsort technique for this step. Heapsort is known to be a very powerful sorting algorithm, requiring only $N \log(N)$ operations to sort a set of N objects. For the present work, the standard *std::sort* algorithm of C++11 was used. It also has a complexity of $O(N \log(N))$. Once the ordered tables are available, the search for contact pairs can begin. Since the master and slave segments are analyzed together, a simple check is necessary to ensure that the detected segments belong to different surfaces, unless the studied problem involves self-contact.



(a) Traditional min/max approach as in Williams and O'Connor (1995)



(b) Adapted pinball-style min/max approach

Figure 4.2: 2D illustration of the sorting global search

The main advantage of using this method is of course the smaller amount of operations required to obtain the list of contact pairs. Another advantage is the storage space required by the algorithm. For N objects in a 3D space, a total of 6 arrays of size N is necessary. The storage requirements are only of order $O(N)$ since the algorithm sorts in place. The spatial sorting algorithm requires more resources for the set up, but is faster in the end due to the considerably smaller amount of checks required. Additionally, since this technique only requires some integer arrays, it does not depend on any special structures and the implementation is straightforward in most programming languages. This method also fulfills the requirement of being robust and indifferent to the geometry of the problem.

Both of the algorithms presented here can also be applied to self-contact problems. It suffices to define both the master and slave as the same body for the global search procedure. In order to prevent unnecessary computations, a filtering process is implemented which automatically excludes the detected contact segments which were also close to each other in the initial configuration.

The detailed spatial sorting procedure including the method used to determine the contact pairs from the ordered lists is best illustrated using an example, which is the topic of the next section.

4.2.2 Spatial Sorting Example

In this section, the general spatial procedure is explained with the help of a simple example. To start off, consider the problem shown in figure 4.2b). Since there are 9 segments in a two-dimensional space, the sorting table is initialized as in table 4.1a). The next step consists of finding the midpoint of the segments and computing their min/max values. For the sake of simplicity, a constant user-specified pinball radius is used, as shown in the picture.

The tables are then sorted using the C++11 *std::sort* algorithm with a custom Boolean function for the comparison. This allows to sort the segment IDs in the table in an ascending order based on their minimum values. This is done for all the principal directions (in this case x and y). After this is done, table 4.1b) is obtained. The $\mathrm{X_{id}}$ and $\mathrm{Y_{id}}$ rows are directly obtained from the sorting algorithm. The $\mathrm{X_{rank}}$ and $\mathrm{Y_{rank}}$ are used as an easy way to find the position of a specific object in the table. For example, if one wants to know how the segment with ID number 5 ranks according to the other segments, a simple look up at index 5 in the $\mathrm{Y_{rank}}$ row shows that it is at position 8 in the $\mathrm{Y_{id}}$ (row of ordered segments along y). The rank rows are filled by a single pass through the id rows. The best way to use these arrays is described in detail in O'Connor (1996). The first step is to choose a primary axis, which in the current case will be x. We then process them object by object, starting from the first position in the id row (in this case: segment number 5). This object will be referenced to as the *pivot*. The next step is to go through the $\mathrm{X_{id}}$ row until the maximal ordinate of the pivot is not larger than the minimal ordinate of the object in the table, i.e. until the Boolean equation 4.5 is true.

$$X_{max,pivot} < X_{min,object} \tag{4.5}$$

The objects which respect this condition are stored in a list of detected segments along x. For this example, this list is $\{1\}$. For every subsequent pivot, only the objects on the right in the table need to be checked (for the primary axis), since all the previous cases have already been processed. The remaining axis (in this case y) is now checked in a similar manner. In

Index	1	2	3	4	5	6	7	8	9
X_{id}	1	2	3	4	5	6	7	8	9
X_{rank}	1	2	3	4	5	6	7	8	9
Y_{id}	1	2	3	4	5	6	7	8	9
Y_{rank}	1	2	3	4	5	6	7	8	9

(a) Before the sorting step

Index	1	2	3	4	5	6	7	8	9
X_{id}	5	1	6	2	7	3	8	4	9
X_{rank}	2	4	6	8	1	3	5	7	9
Y_{id}	1	4	2	3	7	6	8	5	9
Y_{rank}	1	3	4	2	8	6	5	7	9

(b) After the sorting step

Table 4.1: Sorting tables for figure 4.2b)

order to accomplish this, the position of the pivot needs to be found in the Y_{id} row because the search will be starting from this point. This is easily done with the help of the Y_{rank} row, which, as was shown before, returns the position 8. The checks are then performed in both directions of the table because y was chosen to be a secondary axis. The table is processed from the pivot going to the right until equation 4.6 is true. Then, from the pivot to the left until condition 4.7 is true. The objects found during this search are saved in a list of detected segments along y. In the present case, according to figure 4.2b), this list contains the following segment IDs: $\{6, 8, 9\}$.

$$Y_{max,pivot} < Y_{min,object} \tag{4.6}$$

$$Y_{min,pivot} > Y_{max,object} \tag{4.7}$$

However, by proceeding in this manner the lists of detected segments might contain segments belonging to the same surface boundary. This is the case in the y direction. In order to address this issue, a verification is done to ensure that the detected object is from a different surface from that of the pivot, before adding objects to the list of detected segments along y. Of course, this check is not performed in the case of self-contact. The correct list of detected segments along y is therefore an empty set $\{\}$. For three-dimensional problems, the procedure explained here for the y direction is applied analogously for the z direction.

Once the lists of detected segments have been obtained for every direction all that is left to do is a comparison between them to find the intersection. Objects appearing in all of these lists are saved as a contact pair with the pivot. For the current example, there are no contact pair associated with segment number 5. This is due to the small pinball radius and the relative distance between segment number 5 and the segments of the master surface.

4.3 Global Search Comparison

The current section provides a comparison between the two global search algorithms implemented. Using a common setup, the number of contact pairs found along with the time

required to find them is compared. The problem chosen is the Hertz problem in two dimensions. The details concerning the problem can be found in section 7.1. Here, a mesh of 2×2 elements of polynomial order $p = 2$ is used for both bodies. A large value for the interface resolution is chosen. This forces a high number of segments per element from the marching squares algorithm described in section 2.2.5.

The results are listed in table 4.2. Two tests were made using the same setup, only with a different interface resolution. The computation time in milliseconds (ms) corresponds to the time required for the computation and enforcement of the contact constraints at the last time step. As expected, the spatial sorting algorithm is notably faster than the simple pinball algorithm. On average, it requires about 2.5 less time for 2^8 segments per element and 3 times less for 2^9. Moreover, by doubling the number of segments, the pinball algorithm requires about 3.6 times more time to determine the contact pairs. For spatial sorting, this is just around 3 times. It was expected for the pinball radius algorithm to take about 4 times more time due to its complexity of $O(N^2)$. For the spatial sorting, the sorting algorithm itself has a complexity of $O(N \log(N))$. In the current case, by doubling the amount of segments the sorting time should be increased by a factor of about 2.25. However, the computation time observed is 3 times higher. This is due to the fact that the computation time includes not only the sorting, but also the time required to fill the sorting tables and the constraint enforcement. Nonetheless, even if these factors are taken into consideration, the performance of the spatial sorting algorithm is much better than the pinball algorithm.

Another interesting observation is that even if the tests were run on identical setups using the same constant pinball radius for both algorithms, the number of contact pairs found by the spatial sorting technique is greater than for the pinball. This is to be expected considering the differences ensuing from the max/min approach of the spatial sorting algorithm, as shown in figure 4.3. In this figure, a master and a slave body are depicted. The constant ball of radius r is shown for two segments. According to the pinball global search, which measures the distance shown in red in the figure, these two segments should not be stored as a contact pair. However, it respects the conditions of the spatial sorting algorithm and they would therefore be saved. This explains the small discrepancy between the number of contact pairs found by both algorithms. This means that the contact algorithm is run more often using the spatial sorting technique. This is true, however, before computing the expensive contact equations, a projection of points is performed from the slave segment onto the master segment. This allows the determination of a gap or penetration value at different points between the two bodies. If the bodies are not penetrating into each other, the contact pair is disregarded. Basically, in the overall solution procedure, the extra contact pairs found means that more closest-point projections are performed, which are not too computationally expensive especially in two dimensions, as is discussed in chapter 5.

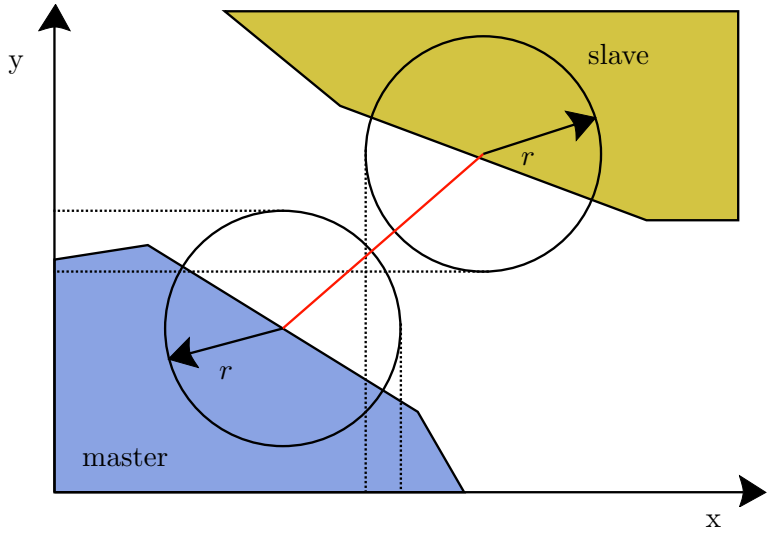


Figure 4.3: Contact constraint computation time versus integration order on the boundary

Load step	**Pinball**		**Spatial Sorting**	
	Computation time (ms)	Contact Pairs	Computation time (ms)	Contact Pairs
1	560.71	250	210.76	271
2	564.77	318	211.46	337
3	564.50	371	220.80	388
4	564.32	413	210.16	435
5	564.28	463	211.22	488
Average	563.72	-	212.88	-

(a) Using 2^8 segments per element

Load step	**Pinball**		**Spatial Sorting**	
	Computation time (ms)	Contact Pairs	Computation time (ms)	Contact Pairs
1	2018	1032	643.16	1167
2	2023	1337	642.22	1466
3	2026	1553	643.17	1680
4	2036	1742	643.84	1877
5	2022	1956	664.93	2086
Average	2025	-	647.46	-

(b) Using 2^9 segments per element

Table 4.2: Comparison of global search algorithms

Chapter 5

Local Search Algorithms

The contact pairs resulting from the chosen global search step are not necessarily in contact. The algorithms presented only ensure that the two segments are within a specified radius of one another. The local search algorithms finds the closest distance between two surfaces through a closest-point projection (CPP) procedure. With the orthogonal projection of a given slave point onto the master surface, it is possible to compute the gap or penetration between the two surfaces, which then serves as a basis for the derivation of the equations for contact. The local search is normally not run on the entire problem at hand as they are way more computationally expensive than the global search. Konyukhov and Izi (2015) gives a general algorithm for both the two- and three- dimensional cases which can be applied to most problem types.

The method used varies depending on the tessellation of the boundary. The marching squares algorithm used in 2D leads to linear segments. In 3D, the marching cubes produces a triangulation of the surface, although planar quadrilaterals can also be defined manually by the user for simple surfaces. The different closest-point projection procedure used for each of these cases is detailed in the present section.

5.1 Projection onto Arbitrary Surfaces

The procedure used for the CPP presented here can be applied to arbitrary surfaces, although it is used for linear quadrilateral surfaces in the current context. The basic idea is to minimize the distance between the projection point and the projected point, as it can be seen in figure 5.1. This can be expressed using the dot product as follows:

$$(\mathbf{r} - \boldsymbol{\rho}(\xi^1, \xi^2)) \cdot (\mathbf{r} - \boldsymbol{\rho}(\xi^1, \xi^2)) \rightarrow \min \tag{5.1}$$

This problem can be seen as an optimization problem where the following functional has to be minimized:

$$\mathcal{F} = \frac{1}{2}(\mathbf{r} - \boldsymbol{\rho}(\xi^1, \xi^2)) \cdot (\mathbf{r} - \boldsymbol{\rho}(\xi^1, \xi^2)) \rightarrow \min \tag{5.2}$$

This minimization problem will be solved using a Newton algorithm. At the minimum, the derivative of $\mathcal{F}$ is necessarily 0 (i.e. $\frac{\partial \mathcal{F}}{\partial \xi^i} = 0$). Therefore, the first and second order derivatives

of $\mathcal{F}$ are required to solve this problem. The first order derivatives are computed as follows:

$$\mathcal{F}' = \begin{bmatrix} \frac{\partial \mathcal{F}}{\partial \xi^1} \\ \frac{\partial \mathcal{F}}{\partial \xi^2} \end{bmatrix} = - \begin{bmatrix} \boldsymbol{\rho}_1 \cdot (\mathbf{r} - \boldsymbol{\rho}) \\ \boldsymbol{\rho}_2 \cdot (\mathbf{r} - \boldsymbol{\rho}) \end{bmatrix} \tag{5.3}$$

And the second derivatives are:

$$\mathcal{F}'' = \begin{bmatrix} \frac{\partial \mathcal{F}}{\partial \xi^1 \partial \xi^1} & \frac{\partial \mathcal{F}}{\partial \xi^1 \partial \xi^2} \\ \frac{\partial \mathcal{F}}{\partial \xi^2 \partial \xi^1} & \frac{\partial \mathcal{F}}{\partial \xi^2 \partial \xi^2} \end{bmatrix} = - \begin{bmatrix} \boldsymbol{\rho}_1 \cdot \boldsymbol{\rho}_1 - \boldsymbol{\rho}_{11} \cdot (\mathbf{r} - \boldsymbol{\rho}) & \boldsymbol{\rho}_1 \cdot \boldsymbol{\rho}_2 - \boldsymbol{\rho}_{12} \cdot (\mathbf{r} - \boldsymbol{\rho}) \\ \boldsymbol{\rho}_2 \cdot \boldsymbol{\rho}_1 - \boldsymbol{\rho}_{21} \cdot (\mathbf{r} - \boldsymbol{\rho}) & \boldsymbol{\rho}_2 \cdot \boldsymbol{\rho}_2 - \boldsymbol{\rho}_{22} \cdot (\mathbf{r} - \boldsymbol{\rho}) \end{bmatrix} \tag{5.4}$$

where the $\boldsymbol{\rho}_i$ and $\boldsymbol{\rho}_{ii}$ are the first and second order derivatives with respect to the surface local coordinates, respectively. The first order derivatives correspond to the tangent vectors at the projected point on the surface and can be obtained directly from the Jacobian of the surface. Since the master segments are planar quads, the second part of the second derivatives $(\boldsymbol{\rho}_{ii} \cdot (\mathbf{r} - \boldsymbol{\rho}))$, which account for the curvature of the surface, are neglected. Now that all the required components are at hand, the following Newton scheme can be set up:

$$\Delta \boldsymbol{\xi}_{(n)} = \begin{bmatrix} \Delta \boldsymbol{\xi}^1_{(n)} \\ \Delta \boldsymbol{\xi}^2_{(n)} \end{bmatrix} = -(\mathcal{F}'')^{-1}_{(n)} \mathcal{F}'_{(n)} \tag{5.5}$$

$$\boldsymbol{\xi}_{(n+1)} = \boldsymbol{\xi}_{(n)} + \Delta \boldsymbol{\xi}_{(n)} \tag{5.6}$$

For the initial guess, the center point of the master segment corresponding to $\xi^1 = \xi^2 = 0$ could be used. To compute a relevant initial guess, a total of 25 points are spread across the surface and their distance with the projection point is evaluated. The closest one to the projection point is taken as a first approximation for the algorithm. More details concerning the solvability of the above equations can be found in Konyukhov and Schweizerhof (2008b).

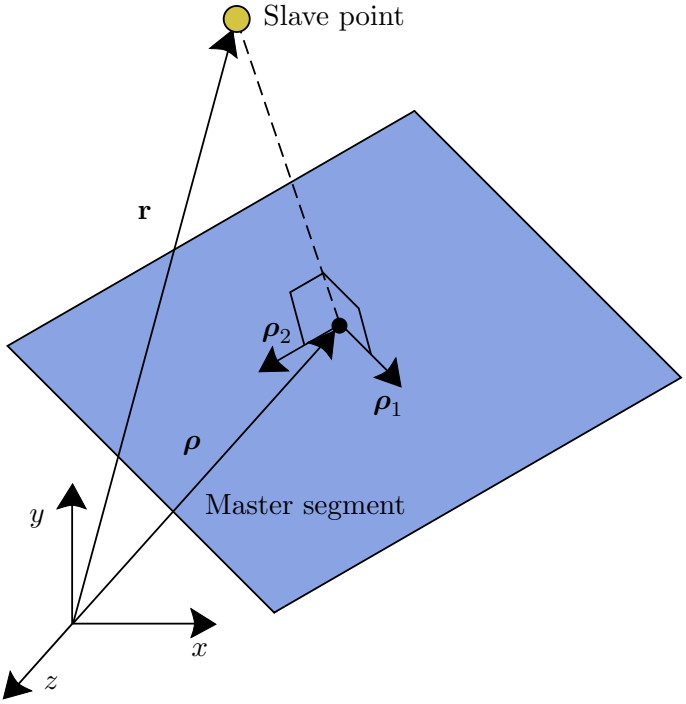


Figure 5.1: Closest-point projection procedure for the 3D case

This algorithm could not be used for triangular segments (as the ones resulting from the marching cubes algorithm) since it requires the Jacobian of the surface and the triangular surfaces are modeled as collapsed quadrilaterals in the code used for this work, which makes the Jacobian equal to 0. The chosen solution presented in section 5.2 turns out to be simpler and computationally less expensive than the iterative procedure presented here, making it an obvious choice.

5.2 Projection onto Triangular Surfaces

The procedure used for the projection on triangular surfaces was proposed by Heidrich (2005). This simple and robust method leads to the barycentric coordinates of the projected point onto the triangle. Let P_1, P_2, P_3 define the three vertices of the triangle and $\mathbf{r}$ the slave point that needs to be projected. The following vectors can be defined:

$$\begin{aligned} \mathbf{u} &= P_2 - P_1 \\ \mathbf{v} &= P_3 - P_1 \\ \mathbf{w} &= \mathbf{r} - P_1 \\ \mathbf{n} &= \mathbf{u} \times \mathbf{v} \end{aligned} \tag{5.7}$$

where $\times$ represents the cross product. The coordinates of the projected point $\boldsymbol{\rho}$ on the master are then:

$$\begin{aligned} \gamma &= \frac{[\mathbf{u}\times\mathbf{w}]\cdot\mathbf{n}}{\mathbf{n}\cdot\mathbf{n}} \\ \beta &= \frac{[\mathbf{w}\times\mathbf{v}]\cdot\mathbf{n}}{\mathbf{n}\cdot\mathbf{n}} \\ \alpha &= 1 - \gamma - \beta \end{aligned} \tag{5.8}$$

This procedure can be visualized in figure 5.2. α, β and γ are the *barycentric* coordinates of the projected point. They are defined as a ratio of areas as

$$\begin{aligned} \alpha &= \frac{A_1}{A_1+A_2+A_3} \\ \beta &= \frac{A_2}{A_1+A_2+A_3} \\ \gamma &= 1 - \alpha - \beta \end{aligned} \tag{5.9}$$

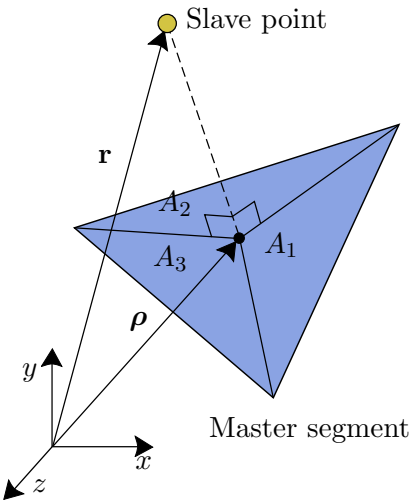


Figure 5.2: Closest-point projection procedure for 3D triangular surfaces

5.3 Projection onto Line Segments

For 2D problems, the object boundaries are, for the purpose of this work, always represented by a set of linear segments. In the contact constraint formulation explained throughout this chapter, points on the slave segments are projected on the master segments. To compute the closest possible on the master segment, a special property of the dot product is used: if two vectors are perpendicular to one another, their dot product equals zero. This procedure is illustrated in figure 5.3. If $\mathbf{s}$ is a vector along the master segment, then $C\mathbf{s}$ can be used to define the master point, where C is a constant. The dot product of the two following vectors is necessarily zero:

$$\mathbf{s} \cdot (\mathbf{r} - C\mathbf{s}) = 0 \Leftrightarrow \mathbf{s} \perp (\mathbf{r} - C\mathbf{s}) \tag{5.10}$$

Using this introduced property, the constant C can be calculated as follows:

$$C = \frac{\mathbf{r} \cdot \mathbf{s}}{\mathbf{s} \cdot \mathbf{s}} \tag{5.11}$$

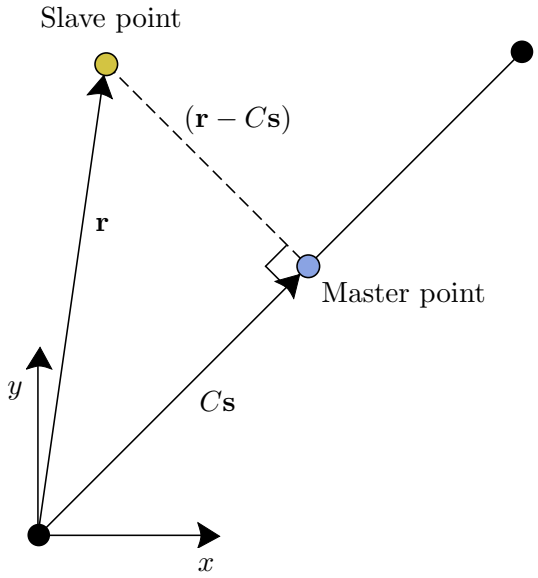


Figure 5.3: Closest-point projection procedure for the 2D case

5.4 CPP Method for Analytically Described Surfaces

Up to this point, the methods presented were assuming a linear approximation of the boundaries. There are obviously some drawbacks associated with such a simplified method. One of the more important is mentioned in Konyukhov and Schweizerhof (2008a): linear approximations of the contact surfaces can lead to oscillations and difficulties in convergence due to the discontinuities (only C^0 continuity) at the inter-element boundaries. Although many authors have proposed techniques for the smoothing of contact surfaces in an effort to reduce the effect of these complications (see Wriggers (2006)), such techniques were deemed unnecessary for the present work, as the error is assumed to be negligible for reasonably small segments.

The procedure described to determine the contact pairs (global search) and the penetration

(local search) has some important drawbacks. The master boundary is approximated by a set of linear segments resulting from the marching squares or marching cubes algorithm presented in section 2.2.5. In order to get the *updated* version of a given segment, an update is required, see section 2.2.6.

A more general option would be to update the exact geometry at every iteration of the Newton scheme to find the projection on the surface. Using an analytical description of the surface, this would lead to a solution which is much closer to reality. However, this method is way more computationally expensive. The tangent vectors also need to be updated to the new configuration in this case. This can be done using the deformation gradient:

$$\boldsymbol{\rho}_i = \frac{\partial X}{\partial x} \cdot \frac{\partial x}{\partial \xi^i} = \mathbf{F} \cdot \mathbf{J} \tag{5.12}$$

where $\mathbf{F}$ is the deformation gradient (see Bonet and Wood (2008)) and $\mathbf{J}$ is the surface Jacobian at the projected point. Another problem would arise also later on during the finite element discretization, where derivatives of the shape functions with respect to the surface convective coordinates are needed (more details in section 6.2.4).

Chapter 6

Penalty regularization of contact constraints

This chapter goes over the concepts behind the regularization of contact constraints using the penalty approach. It focuses on the equations which, based on the contact pairs found, allow for the successful simulation of contact. The first part of this chapter presents the equations required for the simulation of normal contact, from geometry and kinematics to the finite element discretization. Afterwards, the analogous derivations of the formulas for frictional contact are explained.

For the derivation of the necessary equations, a covariant description is used. The chosen finite element discretization type, known as *segment to segment* or STS is easily compatible with this description. The equations derived here follow closely the works of Konyukhov and Schweizerhof (2008a) and Konyukhov and Izi (2015), although some particularities pertinent to embedded interfaces are clearly noted throughout the text.

6.1 Normal Contact

For this first part, only the forces acting along the normal of the contact interface between the two bodies are considered. By doing so, all forces along the tangents are neglected (i.e. frictional forces). Using this assumption, the characteristics of normal contact are no penetration, no shear transfer and of course no adhesion.

This section is divided as follows: an overview of the geometry and kinematics of contact is presented, followed by the weak formulation of the problem. The constitutive equations are discussed before introducing the linearization of the weak form. Finally, the finite element discretization and implementation are given.

6.1.1 Geometry and Kinematics for Normal Contact

One of the surfaces is chosen to be the master surface and convective coordinates are used for the description of the master body. This coordinate system is very well suited for contact problems, as the first two coordinates (ξ^1, ξ^2) describe the surface and the third one (ξ^3) is

perpendicular at every point on the surface, making it aligned with the normal, see figure 6.1. The use of this coordinate system is very well described in Wriggers (2006). The third coordinate can then be used to compute the penetration of the slave body at a given point and the other two to characterize the tangential interaction between the two bodies, as is discussed in the frictional contact section.

The point S on the slave surface corresponds to an integration point obtained from a Gaussian or Lobatto quadrature. The associated point on the master surface C is obtained from a closest point projection procedure as described in chapter 5. Once this point is obtained, one can define the tangent vectors as following:

$$\boldsymbol{\rho}_i = \frac{\partial \boldsymbol{\rho}}{\partial \xi^i} \tag{6.1}$$

with $i = 1, 2$ for the general three-dimensional case. These two vectors form the basis of the tangent place at this point. It is important to note that since they are computed directly from the geometry, they are in general not normalized. From the definition of the cross product, the normal vector can be calculated:

$$\mathbf{n} = \frac{\boldsymbol{\rho}_1 \times \boldsymbol{\rho}_2}{| \boldsymbol{\rho}_1 \times \boldsymbol{\rho}_2 |} \tag{6.2}$$

Using these definitions, the vector $\mathbf{r}$ associated with the slave point $\mathbf{S}$ in figure 6.1 can be described as:

$$\mathbf{r} = \boldsymbol{\rho}(\xi^1, \xi^2) + \xi^3 \mathbf{n} \tag{6.3}$$

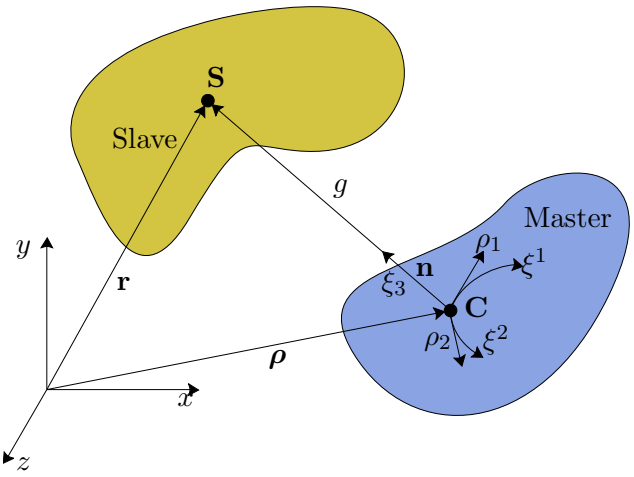


Figure 6.1: Illustration of the main variables for two-body contact

The penetration (or gap) value g can be obtained directly from the third coordinate:

$$g = \xi^3 = (\mathbf{r} - \boldsymbol{\rho}) \cdot \mathbf{n} \tag{6.4}$$

A linearization of the equations is necessary in order to solve the nonlinear system of equations with a Newton-Raphson type algorithm. Even though the problems studied are not transient

(time-dependent), the load increments are considered as velocities. Therefore, the derivatives are computed with respect to a pseudo-time parameter. The full time derivative of the vector $\mathbf{r}$ can be expressed, starting from equation 6.3:

$$\begin{aligned}\frac{d\mathbf{r}(t,\xi^1,\xi^2,\xi^3)}{dt} &= \frac{d\boldsymbol{\rho}(t,\xi^1,\xi^2)}{dt} + \frac{d}{dt}(\xi^3\mathbf{n}(t,\xi^1,\xi^2)) \\ &= \frac{\partial\boldsymbol{\rho}}{\partial t} + \boldsymbol{\rho}_i\dot{\xi}^i + \mathbf{n}\dot{\xi}^3 + \frac{\partial\mathbf{n}}{\partial t}\xi^3 + \frac{\partial\mathbf{n}}{\partial\xi^i}\dot{\xi}^i\xi^3\end{aligned} \tag{6.5}$$

In the previous equation, Einstein's convention has to be applied. It is a simple method to mitigate complex equations. This convention is used extensively in the following derivations. It consists of a basic notation rule to indicate summation over indices. Whenever an index appears twice in a single product term, the summation should be carried out over this index.

The tangential velocity of the master and slave point is defined as:

$$\mathbf{v}_{ma} = \frac{\partial\boldsymbol{\rho}}{\partial t} \tag{6.6}$$

$$\mathbf{v}_{sl} = \frac{\partial\mathbf{r}}{\partial t} \tag{6.7}$$

These definitions can be introduced in equation 6.5 to obtain:

$$\mathbf{v}_{sl} = \mathbf{v}_{ma} + \boldsymbol{\rho}_i\dot{\xi}^i + \mathbf{n}\dot{\xi}^3 + \frac{\partial\mathbf{n}}{\partial t}\xi^3 + \frac{\partial\mathbf{n}}{\partial\xi^i}\dot{\xi}^i\xi^3 \tag{6.8}$$

By evaluating the last equation on the tangent place ($\xi^3 = 0$) and taking the dot product with the tangent vectors $\boldsymbol{\rho}_i$ or the normal vector $\mathbf{n}$ to obtain the rates of deformation for normal and tangential interactions:

$$\dot{\xi}^3 = (\mathbf{v}_{sl} - \mathbf{v}_{ma}) \cdot \mathbf{n} \tag{6.9}$$

$$\dot{\xi}^i = a^{ij}(\mathbf{v}_{sl} - \mathbf{v}_{ma}) \cdot \boldsymbol{\rho}_j \tag{6.10}$$

where a^{ij} is the *contravariant metric tensor*. Its entries can be computed using the tangent vectors obtained from equation 6.1:

$$a_{ij} = \boldsymbol{\rho}_i \cdot \boldsymbol{\rho}_j \tag{6.11}$$

This tensor can be used to switch between the covariant and contravariant descriptions. It contains all the relevant information related to the chosen coordinate system, such as the length of the base vectors and the angle between them.

The contravariant metric tensor is simply the inverse of the covariant, which for a 2×2 matrix can be easily calculated:

$$a^{ij} = \frac{1}{a}\begin{bmatrix} a_{22} & -a_{12} \\ -a_{12} & a_{11} \end{bmatrix} \tag{6.12}$$

where a is the determinant of the covariant metric tensor a_{ij}. The contravariant tangent vectors can now be obtained by multiplying by this metric:

$$\boldsymbol{\rho}^i = a^{ij}\boldsymbol{\rho}_j \tag{6.13}$$

The contravariant and covariant values of the coefficients are obtained in a similar manner.

Subsequent derivations require a second tensor. It is known as the *covariant curvature tensor*. It contains information about, as its name suggests, the curvature of the surface. Its coefficients can be computed as follows:

$$h_{ij} = \boldsymbol{\rho}_{ij} \cdot \mathbf{n} \tag{6.14}$$

where $\mathbf{n}$ is the normal of the surface and $\boldsymbol{\rho}_{ij}$ is the second derivative of the surface vector ρ with respect to the surface coordinates ξ^i. The contravariant coefficients are obtained through multiplication with the metric:

$$h^{ij} = h_{kn} a^{ik} a^{nj} \tag{6.15}$$

In the derivations to follow, the derivatives with respect to time in equations 6.9 and 6.10 are replaced by the variation operator δ. $\dot{\xi}^3$ therefore gives $\delta\xi^3$ and $(\mathbf{v}_{sl} - \mathbf{v}_{ma})$ is changed to $(\delta\mathbf{r} - \delta\boldsymbol{\rho})$.

6.1.2 Weak Formulation

At this point, all the necessary geometrical quantities have been defined. Assuming small values for the penetration g, one can apply the principle of virtual work on the tangent plane at point C. To start with, traction vectors acting on the master ($\mathbf{T}_{ma}$) and the slave ($\mathbf{T}_{sl}$) surfaces are defined. These tractions are acting on an infinitesimal part of the surfaces (dS_{ma} and dS_{sl}). The additivity property of energy allows the consideration of the contact effects independently. The contribution of contact to the principle of virtual work can then be expressed as

$$\delta W_C = \int_{S_{ma}} \mathbf{T}_{ma} \cdot \delta\boldsymbol{\rho}\, dS_{ma} + \int_{S_{sl}} \mathbf{T}_{sl} \cdot \delta\mathbf{r}\, dS_{sl} \tag{6.16}$$

The equilibrium condition in the contact part of the boundary implies that

$$\mathbf{T}_{ma}\, dS_{ma} = -\mathbf{T}_{sl}\, dS_{sl} \tag{6.17}$$

which can now be replaced into equation 6.16 to obtain:

$$\delta W_C = \int_{S_{sl}} \mathbf{T}_{sl} \cdot (\delta\mathbf{r} - \delta\boldsymbol{\rho})\, dS_{sl} \tag{6.18}$$

The above integration is now performed exclusively over the slave surface. From this point on, the variable S is used to designate the slave surface instead of the prior S_{sl} for easier reading. The slave traction vector can be expressed in the local convective coordinates system defined previously on the master surface:

$$\mathbf{T}_{sl} = T_i \boldsymbol{\rho}^i + N\mathbf{n} \tag{6.19}$$

with $i = 1, 2$ representing the tangential tractions along both tangent vectors. Recalling equation 6.8, the variational term in the previous term can be rewritten:

$$(\delta\mathbf{r} - \delta\boldsymbol{\rho}) = \boldsymbol{\rho}_i \delta\xi^i + \mathbf{n}\delta\xi^3 + \delta\mathbf{n}\xi^3 + \frac{\partial\mathbf{n}}{\partial\xi^i}\delta\xi^i\xi^3 \tag{6.20}$$

Substituting equations 6.19 and 6.20 into equation 6.18 and applying the small penetration assumption ($\xi^3 \approx 0$) to evaluate everything on the tangent plane gives

$$\begin{aligned}\delta W_C &= \int_S [N\mathbf{n} + T_j\boldsymbol{\rho}^j][\boldsymbol{\rho}_i\delta\xi^i + \mathbf{n}\delta\xi^3]\, dS \\ &= \int_S N\delta\xi^3 + T_j\boldsymbol{\rho}^j \cdot \boldsymbol{\rho}_i\delta\xi^i\, dS \\ &= \int_S N\delta\xi^3 + T_j\delta_i^j\delta\xi^i\, dS \\ &= \int_S N\delta\xi^3\, dS + \int_S T_i\delta\xi^i\, dS\end{aligned} \tag{6.21}$$

using $\mathbf{n} \cdot \boldsymbol{\rho}_i = 0$ since the two vectors are perpendicular and $\mathbf{n} \cdot \mathbf{n} = 1$ since the normal has unit length. In the derivations above, δ_i^j represents the *Kronecker delta*, which is defined as

$$\delta_i^j = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases} \tag{6.22}$$

At this point, the variations of the convective coordinates are necessary. They can be expressed, on the tangent place, as:

$$\delta\xi^3 = \delta g = (\delta\mathbf{r} - \delta\boldsymbol{\rho}) \tag{6.23}$$

$$\delta\xi^j = a^{ij}(\delta\mathbf{r} - \delta\boldsymbol{\rho}) \cdot \boldsymbol{\rho}_i \tag{6.24}$$

For a detailed derivation of these quantities, the reader is referred to Konyukhov and Schweizerhof (2005b). Replacing equations 6.23 and 6.24 into 6.21 finally results in:

$$\delta W_C = \int_S N(\delta\mathbf{r} - \delta\boldsymbol{\rho}) \cdot \mathbf{n}\, dS + \int_S T_j a^{ij}(\delta\mathbf{r} - \delta\boldsymbol{\rho}) \cdot \boldsymbol{\rho}_i\, dS \tag{6.25}$$

The resulting expression from 6.21 represents the separation of the tangential and normal contributions to the principle of virtual work. It is important to note here that T_j, $j = 1, 2$ are the *covariant* components of the tangential traction.

6.1.3 Constitutive Equations

For the normal part of the traction vector, the constitutive equations result from the Hertz-Signorini-Moreau conditions, see section 3.2.1. From a numerical point of view, the fulfillment of these conditions does not come without difficulties. The method chosen in the course of the present work is based on a penalty regularization of these equations. This allows a certain penetration of the slave point into the master body to occur which depends on a chosen penalty parameter. This leads to an approximate satisfaction of the Hertz-Signorini-Moreau conditions. Consequently, the normal reaction force is expressed in the following way:

$$N = \epsilon_N \langle g \rangle \tag{6.26}$$

where ϵ_N is a normal penalty parameter and $\langle\rangle$ are the *Macaulay* brackets. The latter are used to represent the ramp function:

$$\langle g \rangle = \begin{cases} 0 & \text{if } g > 0 \\ g & \text{if } g \leq 0 \end{cases} \tag{6.27}$$

The normal force is treated as an independent variable in the case of penalty regularization and a rate form (or variational form) is required for the subsequent linearization procedures. This procedure leads to:

$$\dot{N} = \epsilon_N \dot{g} \text{ if } g \leq 0 \tag{6.28}$$

6.1.4 Linearization

The solution process involves a Newtonian method to solve the global equations which first requires a full linearization of the functional obtained in equation 6.25. The full material time derivative is computed for every term of the normal contribution to the virtual work, except the infinitesimal slave surface element dS which is considered constant due to the formulation being based on the master surface local coordinate system. The details of the procedure are omitted here, interested readers are referred to Konyukhov and Schweizerhof (2004) for the exhaustive derivation. The resulting linearized equation is expressed as follows:

$$\begin{aligned} D(\delta W_C^N) = & \int_S \epsilon_N (\delta \mathbf{r} - \delta \boldsymbol{\rho}) \cdot (\mathbf{n} \otimes \mathbf{n})(\mathbf{v}_{sl} - \mathbf{v}_{ma})\, dS \\ & - \int_S N[\delta \boldsymbol{\rho}_{,j} \cdot a^{ij}(\mathbf{n} \otimes \boldsymbol{\rho}_i)(\mathbf{v}_{sl} - \mathbf{v}_{ma}) + (\delta \mathbf{r} - \delta \boldsymbol{\rho}) \cdot a^{ij}(\boldsymbol{\rho}_j \otimes \mathbf{n})\mathbf{v}_{,i}]\, dS \\ & - \int_S N(\delta \mathbf{r} - \delta \boldsymbol{\rho}) \cdot h^{ij}(\boldsymbol{\rho}_i \otimes \boldsymbol{\rho}_j)(\mathbf{v}_{sl} - \mathbf{v}_{ma})\, dS \end{aligned} \tag{6.29}$$

where h^{ij} corresponds to the curvature tensor (see equation 6.15). The comma $_{,i}$ is used to denote a partial derivative with respect to the i^{th} convective coordinate and the $\otimes$ symbol stands for the tensor product. It should be noted that Einstein's summation convention is applied here.

As it can be seen in the resulting equation, the linearization of the covariant formulation leads to a separation of the tangent matrix into three different parts for the normal contribution. The first part of equation 6.29 is known as the main part. It is this part which has the most influence on the emerging tangent matrix. This is due to the fact that the other two are multiplied with the normal force N. This term includes the gap (or penetration) value g which is generally very small. The second part is known as the rotational part and takes into account the rotation of the master body. The third and last part includes the curvature tensor and represents the curvature changes of the master surface.

The linearization process also makes the Macaulay brackets appear ($\langle -g \rangle$) in all three of the terms of equation 6.29. These, however, can be omitted as these equations are applied to contact pairs found through global and local contact search. These procedures ensure a negative gap ($g < 0$) between the master and slave pairs.

6.1.5 Finite Element Discretization and Implementation

The discretization of the contact surface is done using a Segment-to-Segment method described in section 3.4. In order to accomplish this, Gauss points are distributed on the slave segment before being projected on the master surface. This allows a point-wise evaluation of the equations which are then integrated over the slave surface to obtain the global values.

The geometrical boundary required for the STS method is obtained from a marching squares or marching cubes procedure described in section 2.2.5. Due to a total Langrangian formulation, it is necessary to perform an update of the boundary to get the current position, see section 2.2.6. Once the segments have been updated to the current configuration, the integration can take place. The integrals over the slave surface in equation 6.29 are transformed into a sum over the slave integration points:

$$\int_S f(\xi^1, \xi^2)\, dS = \sum_i \sum_j f(\xi_i^1, \xi_j^2) \det J(\eta_i^1, \eta_j^2) w_i w_j \tag{6.30}$$

where ξ_i^1, ξ_j^2 are the local convective coordinates of the point on the master surface obtained from the projection of the slave integration point η_i^1, η_j^2 (local search procedure, see section 5). $\det J(\eta_i^1, \eta_j^2)$ is the determinant of the Jacobian of the transformation $dS \rightarrow d\eta^1 d\eta^2$ (from the slave element surface to the slave local coordinate system). w_i, w_j are the corresponding integration weights according to the Gaussian quadrature.

The implementation of the derived equations requires special attention. The solution vector at every step is of the form:

$$\mathbf{u} = [x_{sl}^1 \cdots x_{sl}^m y_{sl}^1 \cdots y_{sl}^n z_{sl}^1 \cdots z_{sl}^o x_{ma}^1 \cdots x_{ma}^p y_{ma}^1 \cdots y_{ma}^q z_{ma}^1 \cdots z_{ma}^r] \tag{6.31}$$

where the indices $_{sl}$ and $_{ma}$ stand for slave and master, respectively. The variables m, n, o correspond to the total number of degrees of freedom for each independent local coordinate of the slave element. p, q, r are the same for the master element. This allows the use of different discretizations depending on the direction. This can be very useful for certain applications, in particular for problems involving bending. Using a similar notation, the matrix $\mathbf{A}$ is defined:

$$\mathbf{A} = \begin{bmatrix} N_{sl}^1 & \cdots & N_{sl}^m & 0 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & \cdots & 0 & N_{sl}^1 & \cdots & N_{sl}^n & 0 & \cdots & 0 \\ 0 & \cdots & 0 & 0 & \cdots & 0 & N_{sl}^1 & \cdots & N_{sl}^o \\ \end{bmatrix} \\ \begin{bmatrix} -N_{ma}^1 & \cdots & -N_{ma}^p & 0 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & \cdots & 0 & -N_{ma}^1 & \cdots & -N_{ma}^q & 0 & \cdots & 0 \\ 0 & \cdots & 0 & 0 & \cdots & 0 & -N_{ma}^1 & \cdots & -N_{ma}^r \end{bmatrix} \tag{6.32}$$

where the shape functions are evaluated at the Gauss point (η_i^1, η_j^2) for the slave body and at the projection on these Gauss point on the master surface (ξ_i^1, ξ_j^2) for the master body. Because of the total Lagrangian description and the finite cell method concept, the evaluation of the shape functions at the right point can be confusing. Extra care has to be taken to find the right point in the element in the initial configuration. For the slave, the Gauss point local coordinates are known. These can be globalized on the initial slave segment, yielding directly the point where the shape functions need to be evaluated. For the master surface, the procedure is similar. The coordinates obtained from the closest point projection procedure

are located on the *updated* (or current) master segment. This point then needs to be localized on the current segment only to be globalized on the *initial* master segment. The master shape functions can now be evaluated at the right point in the segment's embedding element.

The **A** matrix can also be derived with respect to the master surface local coordinates:

$$\mathbf{A}_{,i} = \begin{bmatrix} 0 & \cdots & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & \cdots & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & \cdots & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ -N^1_{ma,i} & \cdots & -N^p_{ma,i} & 0 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & \cdots & 0 & -N^1_{ma,i} & \cdots & -N^q_{ma,i} & 0 & \cdots & 0 \\ 0 & \cdots & 0 & 0 & \cdots & 0 & -N^1_{ma,i} & \cdots & -N^r_{ma,i} \end{bmatrix} \tag{6.33}$$

where $,i$ corresponds to the partial derivative with respect to the local master segment coordinate ξ^i. The slave shape functions (first part of the above matrix) disappear as they do not depend on these coordinates. In a finite cell method context, due to the geometrical segment and the element being two separate entities, the derivatives of the master shape functions with respect to the local master segment coordinate are obtained by multiplying by the Jacobian of the segment:

$$\begin{aligned} N_{ma,i} &= \frac{\partial N_{ma}}{\partial \xi^i} \\ &= \frac{\partial N_{ma}}{\partial X}\frac{\partial X}{\partial \xi^i} + \frac{\partial N_{ma}}{\partial Y}\frac{\partial Y}{\partial \xi^i} + \frac{\partial N_{ma}}{\partial Z}\frac{\partial Z}{\partial \xi^i} \\ &= \begin{bmatrix} \frac{\partial X}{\partial \xi^1} & \frac{\partial Y}{\partial \xi^1} & \frac{\partial Z}{\partial \xi^1} \\ \frac{\partial X}{\partial \xi^2} & \frac{\partial Y}{\partial \xi^2} & \frac{\partial Z}{\partial \xi^2} \\ \frac{\partial X}{\partial \xi^3} & \frac{\partial Y}{\partial \xi^3} & \frac{\partial Z}{\partial \xi^3} \end{bmatrix} \begin{bmatrix} \frac{\partial N_{ma}}{\partial X} \\ \frac{\partial N_{ma}}{\partial Y} \\ \frac{\partial N_{ma}}{\partial Z} \end{bmatrix} \\ &= \mathbf{J}_{seg} \cdot \frac{\partial N_{ma}}{\partial \mathbf{X}} \end{aligned} \tag{6.34}$$

where X, Y, Z are the global coordinates. The derivatives of the master shape functions with respect to the global coordinates used in this formula can be easily obtained through multiplication with the *element* Jacobian. This result is then used to fill the $\mathbf{A}_{,\mathbf{i}}$ matrix. The tangent vectors $\boldsymbol{\rho}_i$ can be taken directly as rows of the *segment* Jacobian $\mathbf{J}_{seg}$.

Once these quantities have been obtained, the relative displacement vector can be rewritten:

$$\mathbf{r}(\eta^1, \eta^2) - \boldsymbol{\rho}(\xi^1, \xi^2) = \mathbf{A}\mathbf{x} \tag{6.35}$$

The variational form of the above equation gives the virtual displacement vector:

$$\delta\mathbf{r}(\eta^1, \eta^2) - \delta\boldsymbol{\rho}(\xi^1, \xi^2) = \mathbf{A}\delta\mathbf{x} \tag{6.36}$$

The velocity vector is obtained in a similar manner:

$$\mathbf{v}_{sl}(\eta^1, \eta^2) - \mathbf{v}_{ma}(\xi^1, \xi^2) = \mathbf{A}\mathbf{v} \tag{6.37}$$

Taking the derivatives of equations 6.36 and 6.37 with respect to the local master segment

coordinates ξ^1, ξ^2 leads to:

$$-\delta\boldsymbol{\rho}_{,i}(\xi^1,\xi^2) = \frac{\partial \mathbf{A}}{\partial \xi^i}\delta\mathbf{x} = \mathbf{A}_{,i}\delta\mathbf{x} \tag{6.38}$$

$$-\mathbf{v}_{ma,i}(\xi^1,\xi^2) = \frac{\partial \mathbf{A}}{\partial \xi^i}\mathbf{v} = \mathbf{A}_{,i}\mathbf{v} \tag{6.39}$$

It is now possible to rewrite equation 6.29 in a discretized form:

$$\begin{aligned} D(\delta W_C^N) \approx\ & \delta\mathbf{x}\sum_m\sum_n \epsilon_N \mathbf{A}^T(\mathbf{n}\otimes\mathbf{n})\mathbf{A}\mathbf{v}\ \det J(\eta_m^1,\eta_n^2)w_m w_n \\ & -\delta\mathbf{x}\sum_m\sum_n N[\mathbf{A}_{,j}^T a^{ij}(\mathbf{n}\otimes\boldsymbol{\rho}_i)\mathbf{A} + \mathbf{A}^T a^{ij}(\boldsymbol{\rho}_j\otimes\mathbf{n})\mathbf{A}_{,i}]\mathbf{v}\ \det J(\eta_m^1,\eta_n^2)w_m w_n \end{aligned} \tag{6.40}$$

This last formula corresponds the tangent matrix for the normal contact contribution. Einstein's summation convention is of course applied to the repeated indices i, j. The last part of equation 6.29, corresponding to the curvature part, is omitted due to the linear segments used for the approximation of the contact interface. Konyukhov and Schweizerhof (2005a) also states that these equations can in general be omitted with very little loss of efficiency.

Now that the tangent matrix can be computed, the Newton-Raphson method can be applied to solve the non-linear equations. In order to do so, an expression for the residual is needed. It can be obtained by taking the finite element discretization of equation 6.25, for the normal part:

$$\delta W_C^N \approx \mathbf{R}^N = \sum_m\sum_n N\mathbf{A}^T\mathbf{n}\ \det J(\eta_m^1,\eta_n^2)w_m w_n \tag{6.41}$$

The above derivations can also be applied to self-contact problems. An example of a self-contact application along with other numerical examples verifying the equations derived above can be found in chapter 7.

6.2 Tangential Contact

The equations for normal contact presented in the previous section are pretty robust and sufficient for most practical applications. However, some specific problem types involve non-negligible tangential forces. The simplest example might be the sliding of two surfaces against one another. It can be applied to many problems of practical relevance, such as the simulation of the friction forces acting on a brake disk, the interaction between snow and a ski or many manufacturing and forming processes such as rolling or extrusion.

There are two possible states characterizing two contacting surfaces subject to tangential forces: sticking and sliding. These incorporate tangential forces which are treated differently. It is therefore necessary to determine whether the system is sliding or sticking in order to get a good approximation of the forces involved. Fortunately, this can be predicted using simple laws. This section focuses on the equations, in addition to the ones derived in the previous section, to take these phenomena into account.

The additional geometry and kinematic analyses required for tangential contact are presented

first. Then, the weak formulation of the problem, the constitutive equations and the linearization of the weak form are explained. The finite element discretization and implementation are discussed at the end.

6.2.1 Kinematics for Tangential Contact

For the derivations of the frictional contact equations, some additional geometrical components are required. First and foremost, an expression for the relative motion of a point from one load step to another needs to be derived. This value takes on a role similar to that of the normal gap and therefore will be referred to as the tangential gap.

For large deformations in particular, the projection of the slave integration point onto the master surface can change greatly from one load step to the next. In some cases, this projection might end up on a different element of the master surface, depending on the discretization. For this reason, conventional methods to approximate this gap, which were based on a difference in the local coordinates would produce discontinuities and therefore cannot be used. A geometrical interpretation of the gap has to be performed in the global reference frame. This can be expressed by the following equation:

$$\Delta\boldsymbol{\rho} = \boldsymbol{\rho}_{C*}^{n+1} - \boldsymbol{\rho}_{C}^{n+1} \tag{6.42}$$

where $\boldsymbol{\rho}_{C*}^{n+1}$ corresponds to the result of the slave integration point projection on the *current* master surface. $\boldsymbol{\rho}_{C}^{n+1}$ is the location of the *previous* converged point on the master surface, but on the *current* surface. These quantities are illustrated in figure 6.2.

In order to compute this, the algorithm has to store some data related to the previous load step which has to be accessible in the current time step. A cache containing all the necessary data from the previous step is therefore used. This allows the current step to find the previously found master element and local point. The global point corresponding to this local point can then directly be obtained from the saved element, giving the sought-after value $\boldsymbol{\rho}_{C}^{n+1}$.

6.2.2 Constitutive Equations

Considering the tangential forces contribution to the virtual work (second part of equation 6.25) requires an additional constitutive equation. The simplest and most commonly used is the classical Coulomb's law, which was presented in section 3.2.2.

A trial yield function is defined in accordance with Coulomb's friction law to determine whether the system is in the slipping or sliding state:

$$\Phi = \sqrt{a^{ij}T_iT_j} - \mu|N| \tag{6.43}$$

The first term of equation 6.43 corresponds to the L^2 norm of the tangential traction.

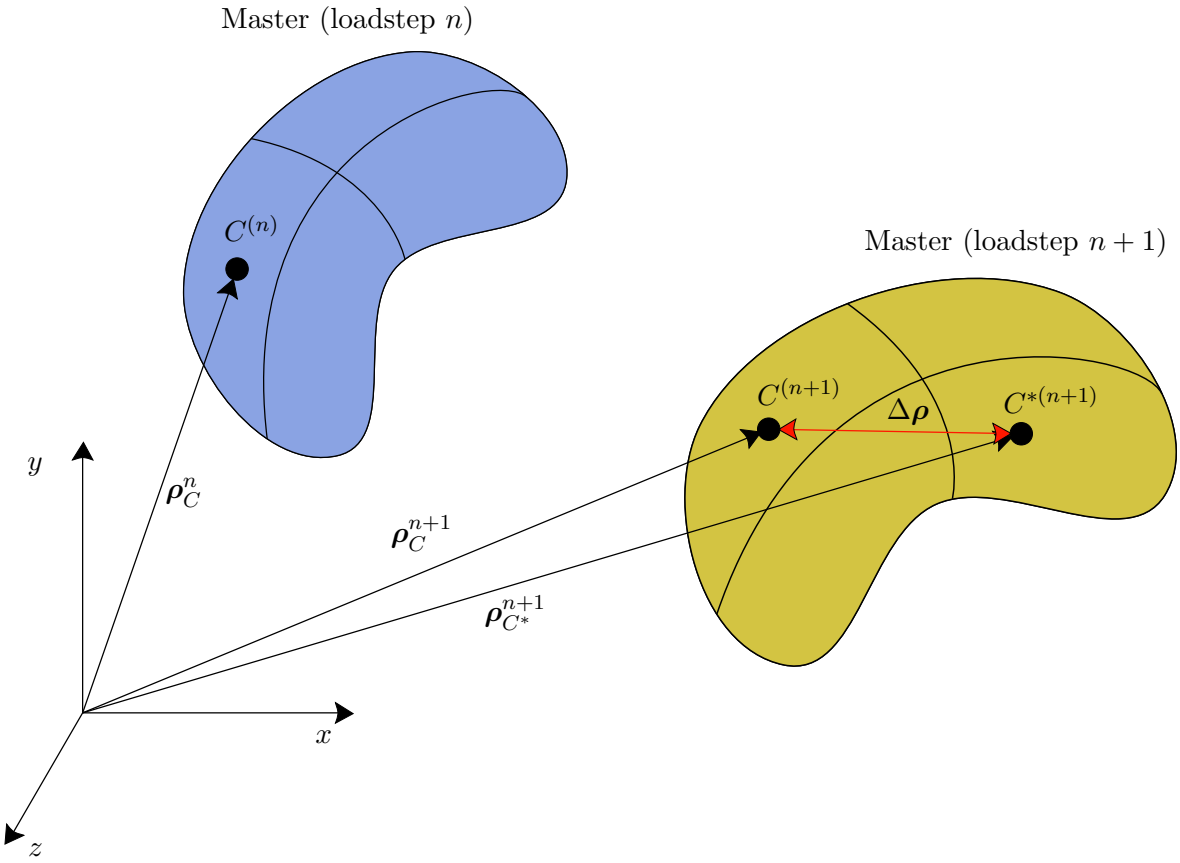


Figure 6.2: Definition of the tangential gap $\Delta\boldsymbol{\rho}$

Konyukhov and Schweizerhof (2005a) proposed to express the tangential traction in the rate form. The derivations in the aforementioned paper lead to the following differential equation:

$$\frac{dT_i}{dt} = (-\epsilon_T a_{ij} + \Gamma^k_{ij} T_k)\dot{\xi}^j - h^k_i T_k \dot{\xi}^3 \tag{6.44}$$

where Γ^k_{ij} are the Christoffel symbols, defined as follows:

$$\Gamma^k_{ij} = \frac{\partial \boldsymbol{\rho}}{\partial \xi^i \partial \xi^j} \cdot \boldsymbol{\rho}^k \tag{6.45}$$

Equation 6.44 is then integrated using a return-mapping algorithm based on the backward Euler implicit scheme. The return-mapping algorithm follows the same principles as the algorithm used in plasticity which bears the same name. In this case, however, the trial step assumes sticking between the two surfaces and the corrector phase maps the tangential tractions back to the sliding case if the values exceed Coulomb's slip surface (defined by the yield function, equation 6.43). This leads to the following return-mapping scheme:

Trial step

$$\begin{aligned} N^{(n+1)} &= \epsilon_N \langle g^{n+1} \rangle \\ (T^{tr})^{n+1}_i &= T^{(n)}_k a^{kj}_{(n)} (\boldsymbol{\rho}^{(n)}_j \cdot \boldsymbol{\rho}^{(n+1)}_i) - \epsilon_T (\Delta \boldsymbol{\rho} \cdot \boldsymbol{\rho_i}) \\ \Phi^{tr}_{(n+1)} &= \sqrt{a^{ij}_{(n+1)} (T^{tr})^{(n+1)}_i (T^{tr})^{(n+1)}_j} - \mu N^{(n+1)} \end{aligned} \tag{6.46}$$

Return mapping

$$T^{n+1}_i = \begin{cases} (T^{tr})^{(n+1)}_i & \text{if } \Phi^{tr}_{(n+1)} \leq 0 \\ \mu N^{(n+1)} \frac{(T^{tr})^{(n+1)}_i}{\|\mathbf{T}^{tr}_{(n+1)}\|} & \text{if } \Phi^{tr}_{(n+1)} > 0 \end{cases} \tag{6.47}$$

These equations contain terms from both the current load step $n+1$ and the previous one n. This means that the solver needs to store certain variables at every load step for future use. These are, along with the master element and local point required for equation 6.42, the tangential traction, the metric and the segment Jacobian (for the surface tangent vectors).

6.2.3 Linearization

The linearization of the tangential part of equation 6.25 is a bit more complex than for the normal part. It leads to two different tangent matrices for sliding and sticking.

For the sticking case, the linearization leads to:

$$\begin{aligned} D(\delta W^{T,st}_C) = &- \int_S \epsilon_T (\delta \mathbf{r} - \delta \boldsymbol{\rho}) \cdot a^{ij} \boldsymbol{\rho}_i \otimes \boldsymbol{\rho}_j (\mathbf{v}_{sl} - \mathbf{v}_{ma})\, dS \\ &- \int_S T^{el}_i [(\delta \mathbf{r} - \delta \boldsymbol{\rho}) \cdot a^{il} a^{jk} \boldsymbol{\rho}_k \otimes \boldsymbol{\rho}_l \mathbf{v}_{,j} + \delta \boldsymbol{\rho}_{,j} \cdot a^{ik} a^{jl} \boldsymbol{\rho}_k \otimes \boldsymbol{\rho}_l (\mathbf{v}_{sl} - \mathbf{v}_{ma})]\, dS \\ &+ \int_S T^{el}_i (\delta \mathbf{r} - \delta \boldsymbol{\rho}) \cdot h^{ij} (\boldsymbol{\rho}_j \otimes \mathbf{n} + \mathbf{n} \otimes \boldsymbol{\rho}_j)(\mathbf{v}_{sl} - \mathbf{v}_{ma})\, dS \end{aligned} \tag{6.48}$$

The interested reader is referred to Konyukhov and Schweizerhof (2005a) for the complete linearization procedure. The first part of equation 6.48 is called the *main* part. Just like for the tangent matrix for normal contact presented in section 6.1.4, the last two parts are known as the rotational part and the curvature part, respectively.

In case of sliding, the following equation is obtained:

$$
\begin{aligned}
D(\delta W_C^{T,sl}) = &- \int_S \frac{\mu\epsilon_N}{\| \mathbf{T}^{el} \|}(\delta\mathbf{r} - \delta\boldsymbol{\rho}) \cdot T_i^{el} a^{ij} \boldsymbol{\rho}_j \otimes \mathbf{n}(\mathbf{v}_{sl} - \mathbf{v}_{ma})\, dS \\
&- \int_S \frac{\epsilon_T \mu |N|}{\| \mathbf{T}^{el} \|}(\delta\mathbf{r} - \delta\boldsymbol{\rho}) \cdot a^{ij} \boldsymbol{\rho}_i \otimes \boldsymbol{\rho}_j(\mathbf{v}_{sl} - \mathbf{v}_{ma})\, dS \\
&+ \int_S \frac{\epsilon_T \mu |N|}{\| \mathbf{T}^{el} \|^3}(\delta\mathbf{r} - \delta\boldsymbol{\rho}) \cdot T_i^{el} T_j^{el} a^{ik} a^{jl} \boldsymbol{\rho}_k \otimes \boldsymbol{\rho}_l(\mathbf{v}_{sl} - \mathbf{v}_{ma})\, dS \\
&- \int_S \frac{\mu |N| T_i^{el}}{\| \mathbf{T}^{el} \|}[(\delta\mathbf{r} - \delta\boldsymbol{\rho}) a^{il} a^{jk} \boldsymbol{\rho}_k \otimes \boldsymbol{\rho}_l \mathbf{v}_{,j} + \delta\boldsymbol{\rho}_{,j} a^{ik} a^{jl} \boldsymbol{\rho}_k \otimes \boldsymbol{\rho}_l(\mathbf{v}_{sl} - \mathbf{v}_{ma})]\, dS \\
&+ \int_S \frac{\mu |N| T_i^{el}}{\| \mathbf{T}^{el} \|} h^{ij}(\delta\mathbf{r} - \delta\boldsymbol{\rho}) \cdot (\boldsymbol{\rho}_j \otimes \mathbf{n} + \mathbf{n} \otimes \boldsymbol{\rho}_j)(\mathbf{v}_{sl} - \mathbf{v}_{ma})\, dS \\
&- \int_S \frac{\mu |N|}{\| \mathbf{T}^{el} \|^3}(\delta\mathbf{r} - \delta\boldsymbol{\rho}) \cdot T_{el}^i T_{el}^k T_{el}^n \Gamma_{n,kj} a^{jm} \boldsymbol{\rho}_i \otimes \boldsymbol{\rho}_j(\mathbf{v}_{sl} - \mathbf{v}_{ma})\, dS \\
&+ \int_S \frac{\mu |N|}{\| \mathbf{T}^{el} \|^3}(\delta\mathbf{r} - \delta\boldsymbol{\rho}) \cdot T_{el}^i T_{el}^j T_{el}^k h_{jk} \boldsymbol{\rho}_i \otimes \mathbf{n}(\mathbf{v}_{sl} - \mathbf{v}_{ma})\, dS
\end{aligned}
\tag{6.49}
$$

where $\mathbf{T}^{el}$ represents the trial elastic tangential traction from equation 6.46. It is important to note the appearance of both covariant (T_i^{el}) and contravariant (T_{el}^i) components of the tangential tractions. It is possible to change from one to the other using the metric, see equation 6.13. For this equation, the first three terms constitute the main part and the fourth one corresponds to the rotational part. The last three are related to the changes in curvature of the surface. The equation shown here has been shortened using Einstein's summation convention.

As for the normal case, the curvature terms are neglected for the present work even though high order elements are used. This is due to the combination of the marching squares (or cubes) along with the local search technique used, see section 2.2.5 and chapter 5.

6.2.4 Finite Element Discretization and Implementation

The finite element discretization of the equations developed thus far can be done in various ways. An overview of the different possibilities was introduced in section 3.4. The present section focuses on the discretized version of equations 6.48 and 6.49.

For the sticking case:

$$D(\delta W_C^{T,st}) \approx -\delta\mathbf{x}\sum_m\sum_n \epsilon_T \mathbf{A}^T a^{ij}(\boldsymbol{\rho}_i\otimes\boldsymbol{\rho}_j)\mathbf{A}\mathbf{v}\ \det J(\eta_m^1,\eta_n^2)w_m w_n$$
$$-\delta\mathbf{x}\sum_m\sum_n T_i^{el}[\mathbf{A}^T a^{il}a^{jk}(\boldsymbol{\rho}_k\otimes\boldsymbol{\rho}_l)\mathbf{A}_{,j}+\mathbf{A}_{,j}^T a^{ik}a^{jl}(\boldsymbol{\rho}_k\otimes\boldsymbol{\rho}_l)\mathbf{A}]\mathbf{v}\ \det J(\eta_m^1,\eta_n^2)w_m w_n \tag{6.50}$$

And the sliding case:

$$D(\delta W_C^{T,sl}) \approx -\delta\mathbf{x}\sum_m\sum_n \frac{\mu\epsilon_N}{\parallel\mathbf{T}^{el}\parallel}\mathbf{A}^T T_i^{el}a^{ij}\boldsymbol{\rho}_j\otimes\mathbf{n}\mathbf{A}\mathbf{v}\ \det J(\eta_m^1,\eta_n^2)w_m w_n$$
$$-\delta\mathbf{x}\sum_m\sum_n \frac{\epsilon_T\mu|N|}{\parallel\mathbf{T}^{el}\parallel}\mathbf{A}^T a^{ij}\boldsymbol{\rho}_i\otimes\boldsymbol{\rho}_j\mathbf{A}\mathbf{v}\ \det J(\eta_m^1,\eta_n^2)w_m w_n$$
$$+\delta\mathbf{x}\sum_m\sum_n \frac{\epsilon_T\mu|N|}{\parallel\mathbf{T}^{el}\parallel^3}\mathbf{A}^T T_i^{el}T_j^{el}a^{ik}a^{jl}\boldsymbol{\rho}_k\otimes\boldsymbol{\rho}_l\mathbf{A}\mathbf{v}\ \det J(\eta_m^1,\eta_n^2)w_m w_n$$
$$-\delta\mathbf{x}\sum_m\sum_n \frac{\mu|N|T_i^{el}}{\parallel\mathbf{T}^{el}\parallel}[A^T a^{il}a^{jk}\boldsymbol{\rho}_k\otimes\boldsymbol{\rho}_l\mathbf{A}_{,j}+\mathbf{A}_{,j}^T a^{ik}a^{jl}\boldsymbol{\rho}_k\otimes\boldsymbol{\rho}_l\mathbf{A}]\mathbf{v}\ \det J(\eta_m^1,\eta_n^2)w_m w_n \tag{6.51}$$

As for the normal contact case, a residual is needed for the tangential contact equations to go in par with the above defined tangent matrices. It can be obtained from the finite element discretization of the second part of equation 6.25:

$$\delta W_C^T \approx \mathbf{R}^T = \sum_m\sum_n T_j a^{ij}\mathbf{A}^T\boldsymbol{\rho}_i\ \det J(\eta_m^1,\eta_n^2)w_m w_n \tag{6.52}$$

These expressions for the residual and the tangent matrices then have to be combined with the ones developed for normal contact presented in the previous section.

Chapter 7

Numerical Results

A total of four examples are presented here to demonstrate the functionalities of the algorithms derived in the previous chapters. To assess the performance of the implementation, the numerical results are compared with analytical solution for normal contact. The examples have been specifically chosen to illustrate specific capabilities, such as large deformation, self-contact, 2D and 3D simulation using the finite cell method. The simulation of friction, however, is not covered in the following examples.

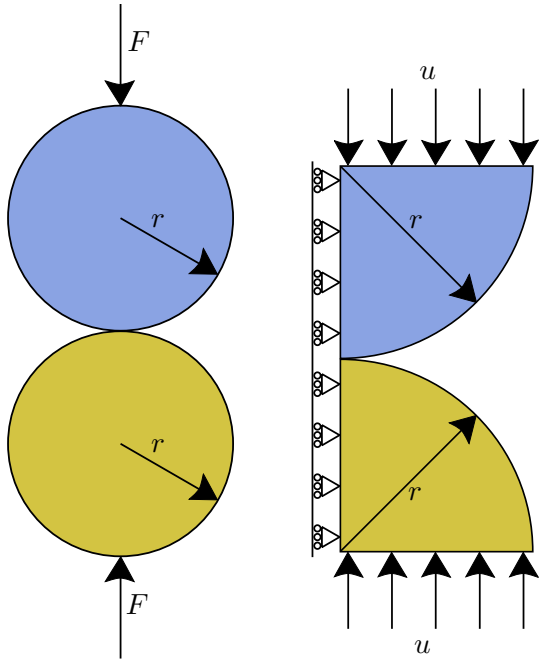
7.1 2D Hertz Problem

Probably the simplest problem in contact mechanics is the contact between two infinitely long cylinders. The first study on this problem was published by Hertz in 1881 (Hertz, 1882). It is one of the few problems in the field of contact mechanics for which an analytical solution exists. Therefore, it is used extensively to assess the quality of different algorithms and formulations. Hertz used many assumptions for his solution: linear elasticity, frictionless contact surfaces, isotropic and homogeneous material and a contact surface which is C^2-smooth which is much smaller than the bodies. For more information on the analytical solution, the reader is referred to Franke (2011) and the references therein.

The two cylinder problem can be reduced to a two-dimensional plane strain problem involving two circles as in figure 7.1a). Using symmetry, this can be further simplified to obtain the more computationally friendly problem shown in figure 7.1b). Franke (2011) has performed extensive analysis on a contact formulation for high-order elements using two different setups: bilateral quarter circle as in figure 7.1b) and quarter circle against a rigid planar surface. These two models should yield equivalent results. In the present work, a different formulation is used and the analysis is done using the finite cell method presented in chapter 2. The boundaries of both surfaces are recuperated using the marching squares algorithm (see 2.2.5).

The parameters used for the current study are the following:

$$
\begin{array}{lcllcl}
E & = & 1.0 \times 10^5 & u & = & -0.1 \\
\nu & = & 0.3 & \alpha & = & 1.0 \times 10^{-10} \\
r & = & 4 & \epsilon_N & = & 1.0 \times 10^7 \\
\text{gap} & = & 0.0 & & &
\end{array}
$$



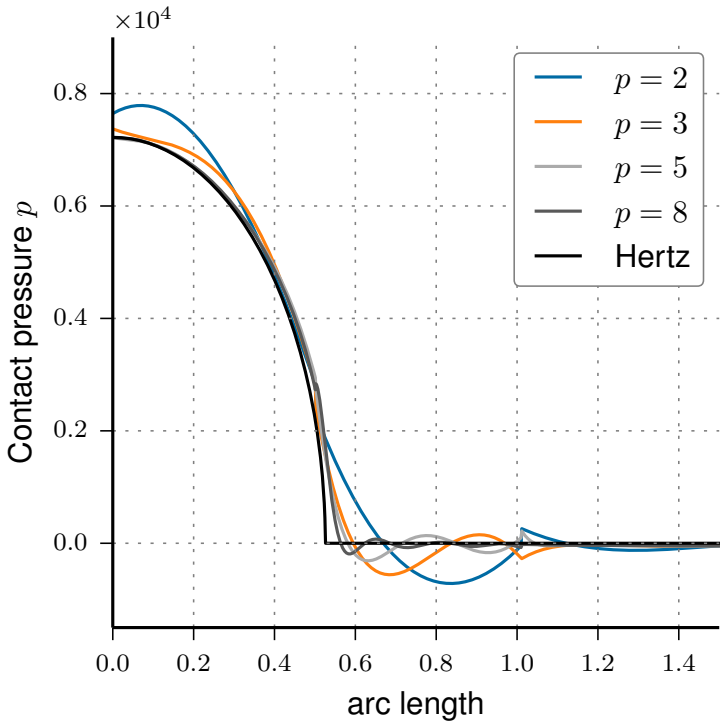
(a) Simplification to a plane strain problem

(b) Problem setup for analysis

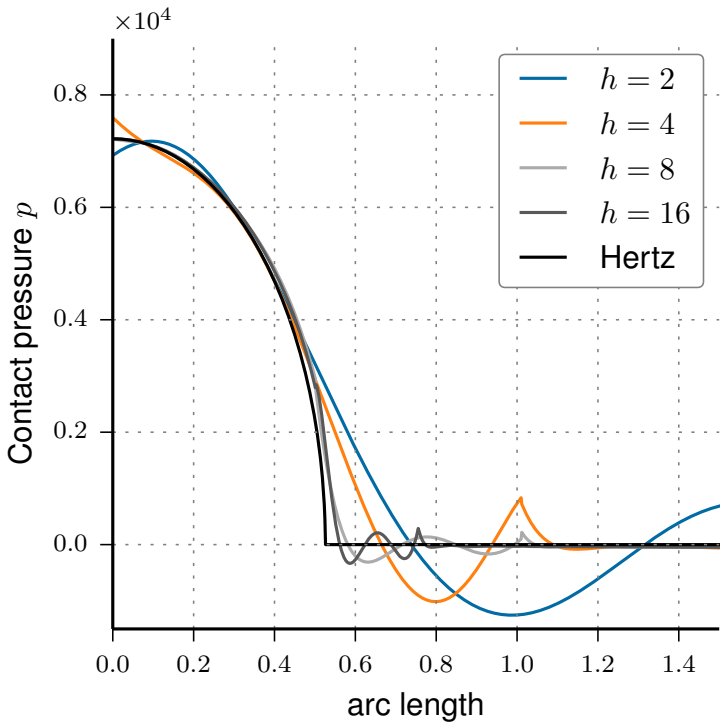
Figure 7.1: The Hertzian problem in two dimensions

Figure 7.2 shows a comparison of the results with the analytical solution of the Hertz problem. The effect of the polynomial order of the shape functions on the solution, or *p-study*, and the effect of mesh refinement, known as *h-study* are illustrated in two separate plots. In these graphs, the pressure is plotted against the arc length of the top quarter circle, which was defined as the slave body for in the analysis process.

Figure 7.2a) presents the results for 8×8 finite cell elements for both quarter circles for different polynomial degrees of the shape functions. It can be noted that lower polynomial orders such as 2 or 3 cause a slightly overestimated maximum stress. Oscillations of higher amplitude are also observed outside the contact zone. These oscillations are a well-known and documented problem resulting from the use of high-order elements, as it was reported in Franke et al. (2010). As the polynomial order is increased to 5 and 8, the results become hardly distinguishable from the analytical solution, especially near the point of maximum pressure. The oscillations outside of the contact zone are reduced and the kink in the stresses at the limit of the contact zone is better represented. A definitive convergence to the analytical solution is observed as the polynomial order is increased. For the h-study, a polynomial order of 5 is used for the shape functions of both meshes. The number of elements is subsequently increased from 2×2 to 4×4, 8×8 and 16×16. The results along with the analytical solution are shown in figure 7.2b). The results are, for all meshes, very close to the analytical solution near the peak pressure. This can be explained by the high polynomial order. However, oscillations appear outside of the contact zone for coarser meshes. These are significantly reduced when passing from 4×4 to 8×8 elements, partly due to a node in the 8×8 and 16×16 cases being positioned close to the end of the contact zone. Franke (2011) has shown that meshes without a node at the contact zone interface results in greater oscillations, due to the fact that it implies a change in boundary conditions along one edge.



(a) Influence of the polynomial order for meshes of 8×8 elements



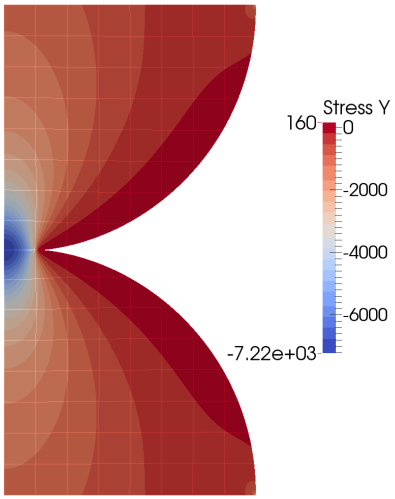
(b) Influence of the number of elements for $p = 5$

Figure 7.2: Comparison of the results with the analytical solution for the Hertz problem

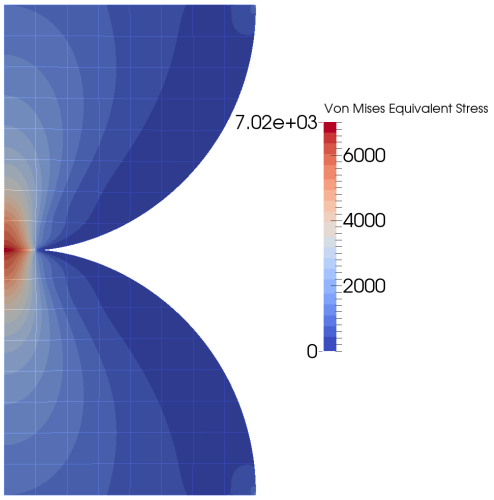
The results show that a relatively fine recovery of the boundary using linear segments from the marching squares algorithm leads to very good approximations of the solution. In terms of implementation complexity, this technique is much simpler than the methods proposed by Konyukhov et al. (2015). Moreover, it is not problem-specific and provides more accurate results for similar parameters.

The stress distribution obtained from the analysis using 8×8 meshes and a polynomial order of 5 are shown in figure 7.3. The stress distribution is perfectly symmetric between the master and the slave and is coherent with what was obtained in previous works. The shear stresses in 7.3d) might seem different at first glance, but this is only due to the difference in sign between the master and slave. The stress scale is adapted to the shear stresses on the master surface (bottom quarter circle) which is why they appear much less in the slave counterpart. As it was expected, the maximum in shear stress is observed at the limit of the contact zone.

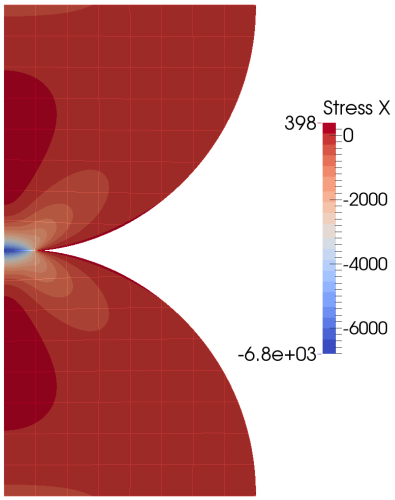
To further improve the results, a promising way would be to use adaptivity techniques. There are many techniques available, such as mesh refinement, polynomial degree refinement, node relocation or hierarchical refinement, as described in Zander et al. (2015). However, this is beyond the scope of this work.



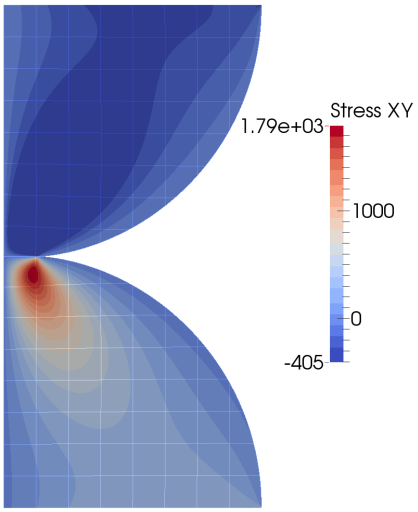
(a) Stresses in Y



(b) Von Mises equivalent stress



(c) Stresses in X



(d) Shear stresses

Figure 7.3: Stress distribution for the Hertz problem with $h = 8 \times 8$ and $p = 5$

7.1.1 Effects of Under Integration

In order to optimize the performance of the algorithm proposed in this work, a study was conducted to determine the most computationally demanding functions. The two-dimensional Hertz problem presented in the current section was used as a benchmark. The Segment-to-Segment approach requires the computation of the contact equations for every quadrature point on the slave boundary, as discussed in section 6.1.5. Therefore, reducing the integration order for the Gaussian quadrature diminishes the number of times the contact equations are computed, as well as the overall time for the numerical integration scheme.

The tests were performed using a constant integration order, independent of the polynomial order of the ansatz functions used. The results for an ansatz order of $p = 5$ and two meshes of 2×2 elements are shown in table 7.1. The table shows the time in milliseconds required for the computation of the contact constraints at the last load step of the simulation. This includes the time required for the computation of the contact equations as well as the integration itself. The plot in figure 7.4 shows the values from table 7.1 along with a linear regression with a R^2 coefficient of 0.96. It confirms the expectation that the time required is reduced linearly with respect to the integration order. This is due to the linear relation between the number of integration points for the Gaussian quadrature and the integration order and, consequently, the number of times the contact algorithm is run. This is valid for two-dimensional problems, since the boundaries are in one dimension and therefore the Gaussian quadrature for one dimension is used. For three-dimensional problems, a quadratic relation is expected, as the number of Gauss integration points decreases in a quadratic manner with respect to the integration order for the two-dimensional boundary components.

For the $p = 5$ case, a reduction of about 52% in the computation time for the contact constraints was obtained, which corresponds to a difference of about 2.75% in the total simulation time. Furthermore, almost no change in the computed maximum stresses was observed. This shows that a reasonably fine recovery of the boundary using the marching squares algorithm leads to plausible results, without the need for a complete Gaussian integration.

	Computation time (ms)	**Maximum stress (Mpa)**
Int. Order = 8 (Ansatz+3)	471.19	1200.28
Int. Order = 7 (Ansatz+2)	460.07	1200.28
Int. Order = 6 (Ansatz+1)	421.80	1200.29
Int. Order = 5 (Ansatz+0)	367.27	1200.28
Int. Order = 4 (Ansatz-1)	343.74	1200.28
Int. Order = 3 (Ansatz-2)	335.99	1200.29
Int. Order = 2 (Ansatz-3)	304.10	1200.29
Int. Order = 1 (Ansatz-4)	225.12	1199.42

Table 7.1: Results for the under integration study of the 2D Hertz problem

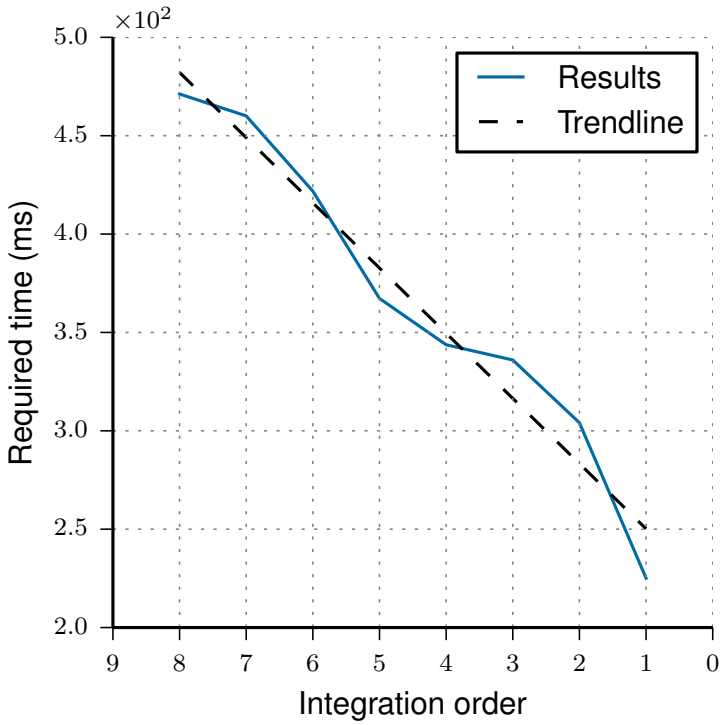


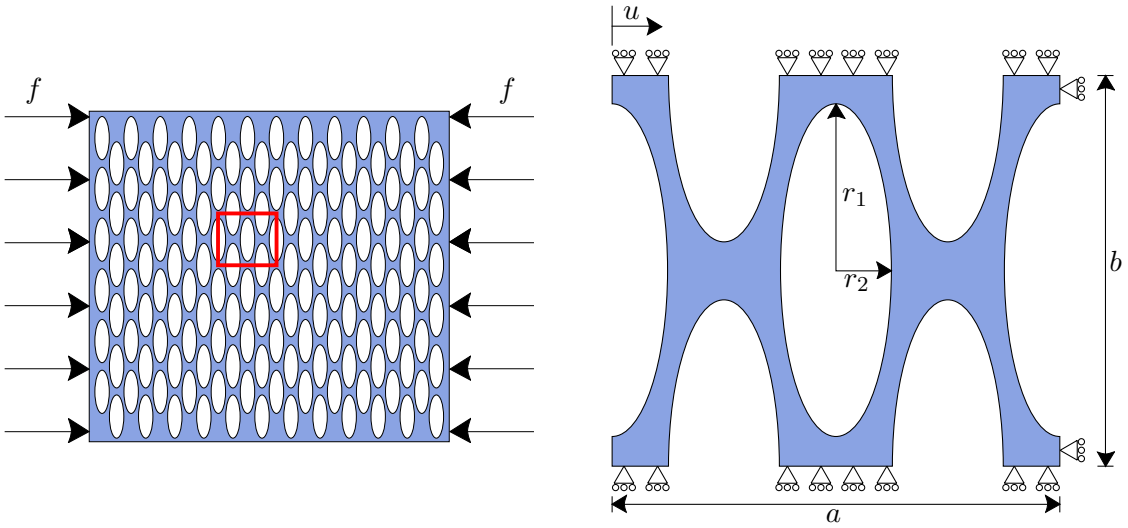
Figure 7.4: Contact constraint computation time versus integration order on the boundary

7.2 Compression of a Foam

The example presented here corresponds to the extreme compression of a foam material. It showcases the large deformation capabilities of the finite cell Method along with the self-contact possibilities of the formulation used in this work. A general model for a perfect foam is shown in figure 7.5a), from which a unite cell delimited by the red box is taken for the current analysis. Applying the necessary boundary condition and the symmetry, the setup for the current problem can be obtained as in figure 7.5b).

For the current analysis, a FCM mesh of 7 elements in the y direction and 12 element in the x direction with a polynomial order of $p = 3$ is used. The stresses are computed using the Neo-Hooke material law introduced in section 2.2.4. The parameters used for the simulation are the following, referring to 7.5b):

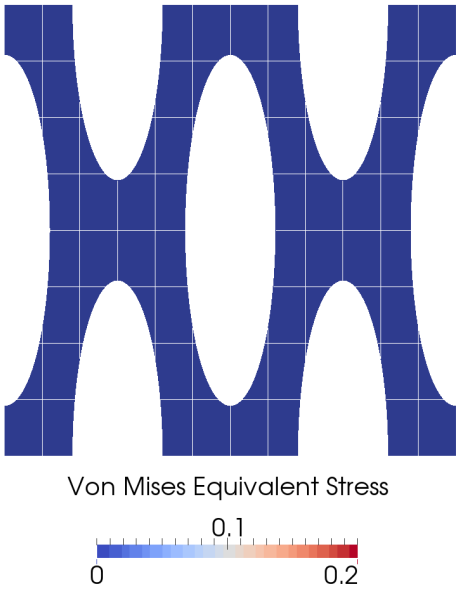
$$\begin{aligned} E &= 1.0 \\ \nu &= 0.3 \\ a &= 8 \\ b &= 8 \end{aligned} \qquad \begin{aligned} r_1 &= 28/9 \\ r_2 &= 0.8 \\ \epsilon_N &= 1000 \\ u_{max} &= 3.15 \end{aligned}$$



(a) Ideal porous material model **(b)** Porous material problem setup for analysis

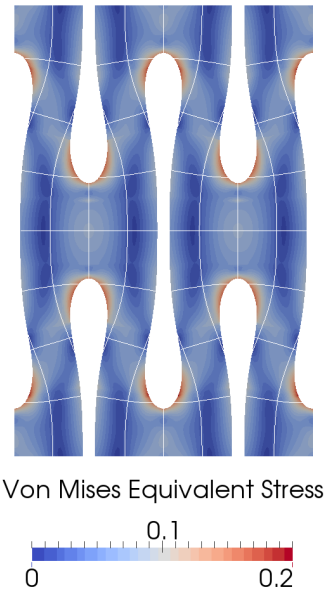
Figure 7.5: Compression of a porous material

Figure 7.6 shows the resulting von Mises stress distribution for several load steps. As it can be seen in the last picture, the displacement leads to a complete crush of the foam structure. The von Mises stress distribution is coherent with what was expected, with the highest stress concentration located at the tip of the ellipse, where the biggest material distortion is observed. The stresses in this region are extremely high due to the absence of plasticity in the Neo-Hookian material formulation used. This also causes some instability and therefore some difficulties in convergence in the last time steps.

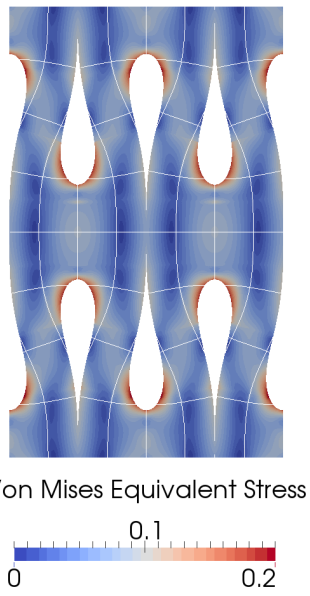


(a) At displacement $u = 0$

(b) At displacement $u = 0.43 \times u_{max}$



(c) At displacement $u = 0.86 \times u_{max}$



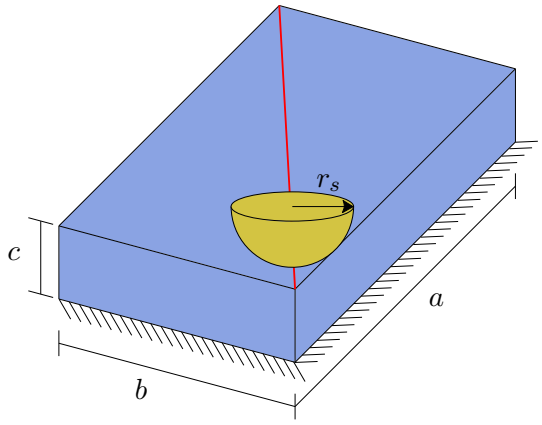
(d) At displacement $u = u_{max}$

Figure 7.6: Von Mises equivalent stress at different load steps for the compression of a porous material

7.3 3D Large Sliding

The third example consists of the sliding of a sphere on the top surface of a block. It demonstrates the proper functioning of the global and local search as well as the constraint enforcement with surfaces which are not obtained from a marching cubes algorithm. One of the surfaces is instead constructed as a set of regular quadrilaterals.

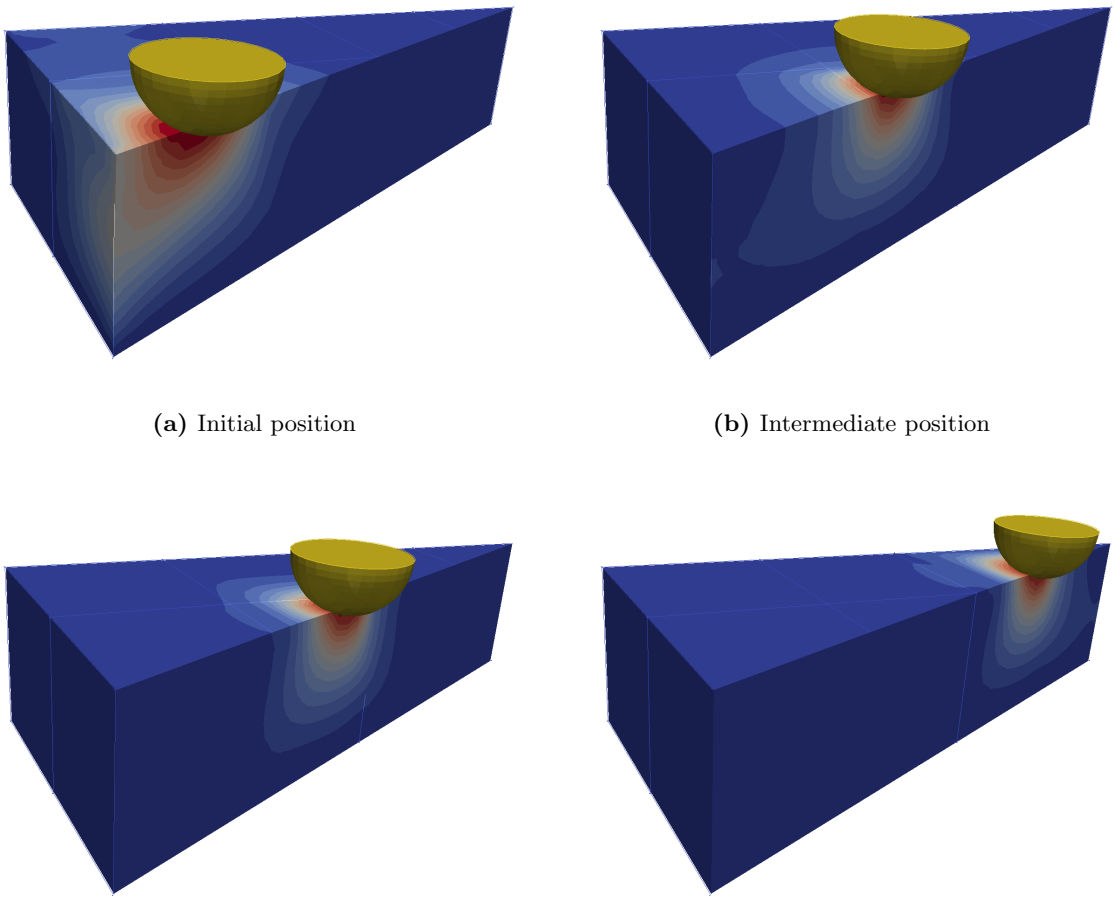
The problem definition as well as the parameters used is shown in figure 7.3. The problem uses two independent meshes: a high-order finite element mesh of $4 \times 2 \times 1$ elements for the block depicted in blue and a finite cell mesh of $1 \times 1 \times 1$ element for the sphere. The indices $_s$ and $_b$ refer to the parameters used for the sphere and the block, respectively. For the simulation, the sphere, which is much stiffer, is first pressed onto the block and is afterwards dragged along the diagonal depicted as a red line in figure 7.3. The bottom of the cuboid is fixed in all three directions. The gap value in the parameters corresponds to the vertical displacement value maintained on the sphere as it is sled across the block. This forces a constant vertical load on the block and the sphere.



E_s	=	1.0×10^4
ν_s	=	0.3
p_s	=	1
r_s	=	0.35
gap	=	-0.05
E_b	=	1.0
ν_b	=	0.3
p_b	=	3
a	=	4.0
b	=	2.0
c	=	1.0
ϵ_N	=	200

Figure 7.7: Setup for the 3D Large sliding problem

Figure 7.8 shows the displacement magnitude in the block at four different steps. Figure 7.8a) corresponds to the beginning of the simulation, after the sphere has been pressed against the cuboid. The illustrations that follow present the results obtained at different steps as the sphere is dragged along the block. The block has been cut along its diagonal in order to show the displacements inside the volume. Since the problem can be seen as an extension to three dimensions of the Hertz problem presented in section 7.1, the solutions for both problems can be compared. Although some difficulties in convergence were observed in the solution process, the displacement distribution is analogous between the two examples, which indicates that the algorithm is working properly for this particular type of problems.



(a) Initial position

(b) Intermediate position

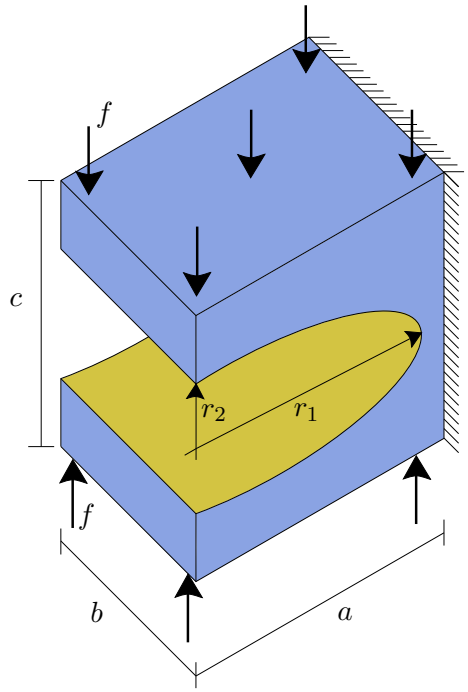
(c) Intermediate position

(d) Final position

Figure 7.8: Results at different load steps for the 3D sliding problem

7.4 3D Large Deformation Self-Contact

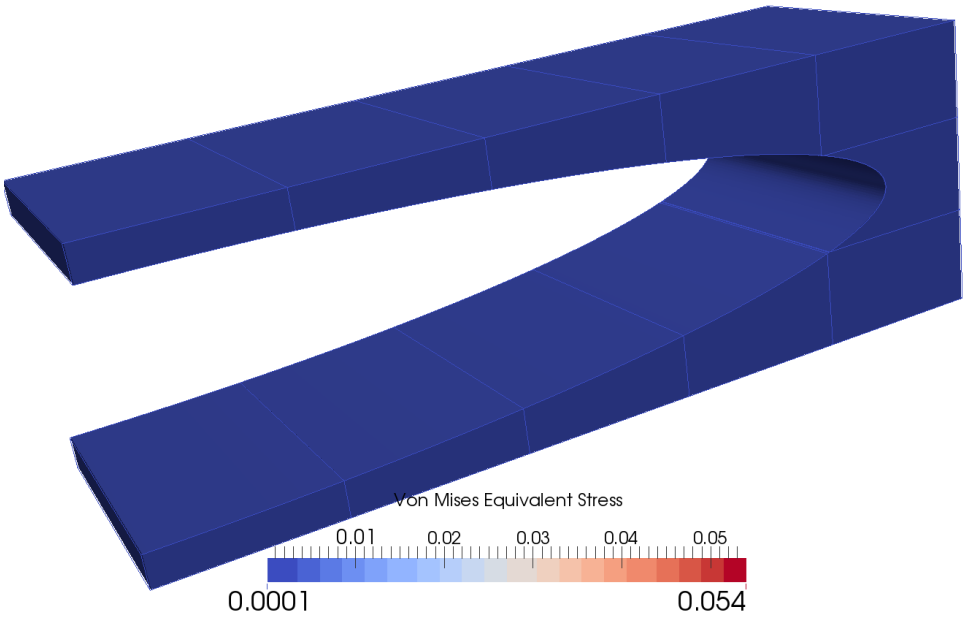
The present example consists of the severe clamping of a three-dimensional body. It serves as a test to ensure the good functionality of three-dimensional self-contact. The problem setup is illustrated in figure 7.4. The finite cell mesh for this problem consists of 5 elements in the X direction, 1 in the Y direction and 3 in the Z directions. The parameters used for the simulation are as follows:



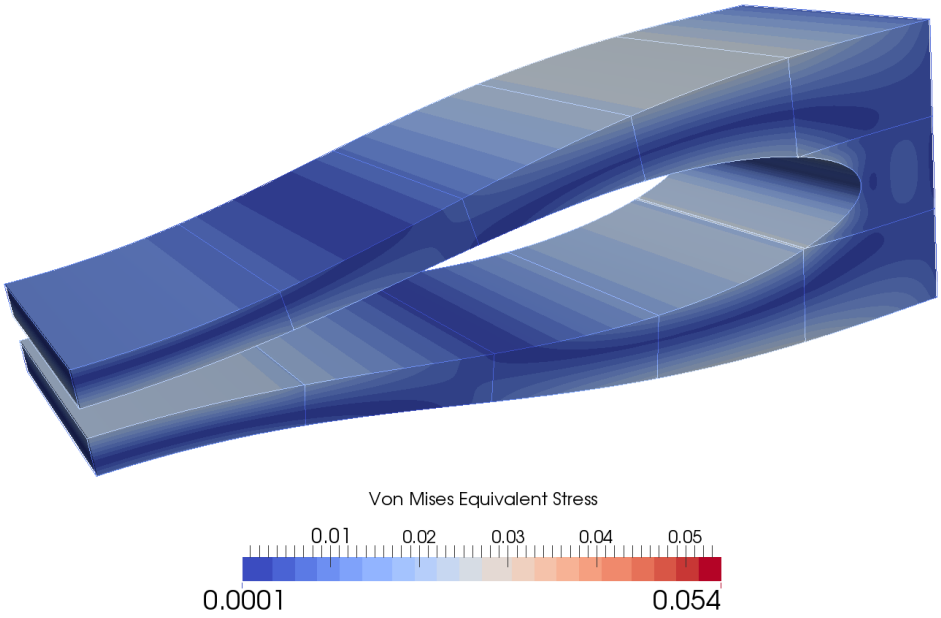
$$
\begin{aligned}
E &= 1.0 \\
r_1 &= 24/18 \\
\nu &= 0.3 \\
r_2 &= 7/36 \\
a &= 1.5 \\
b &= 0.5 \\
c &= 0.5 \\
\epsilon_N &= 1000 \\
f_{max} &= 1.0 \times 10^{-3} \\
p &= 4
\end{aligned}
$$

Figure 7.9: Setup for the 3D self-contact problem

Figures 7.10 and 7.11 present the results as the applied force is slowly increased. The stress distribution is coherent with what was also obtained in two dimensions for the two-dimensional porous material problem presented in section 7.2. The self-contact is detected properly and the general deformation of the body corresponds to the expectations. An equivalent problem was defined on the commercial FEM software Ansys. The mesh used consists of 53 924 linear elements, for which the results are shown in figure 7.12. The first thing that was noted is the variation of the stresses in the thickness direction, which does not appear in the results of the implemented algorithm. There is also a small difference in the maximum stresses. The overall stress distribution, however, is very similar in both analyses, which leads to the conclusion that the results obtained in the course of the current example are a good approximation of the exact solution.

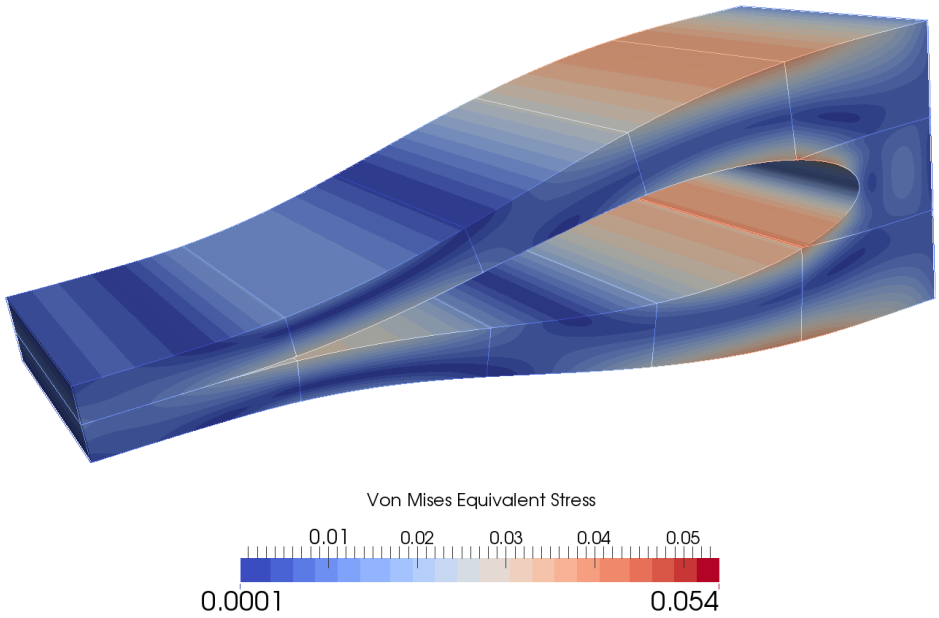


(a) At $f = 0$

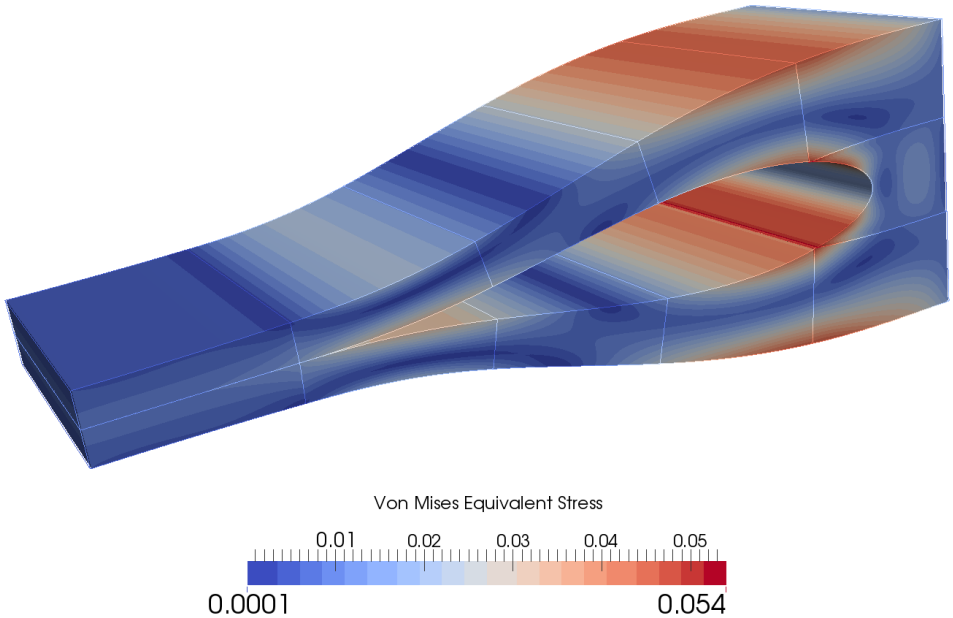


(b) At $f = 0.2 \times f_{max}$

Figure 7.10: Equivalent von Mises stress for the 3D self-contact problem under low loads



(a) At $f = 0.6 \times f_{max}$



(b) At $f = f_{max}$

Figure 7.11: Equivalent von Mises stress for the 3D self-contact problem under important loading

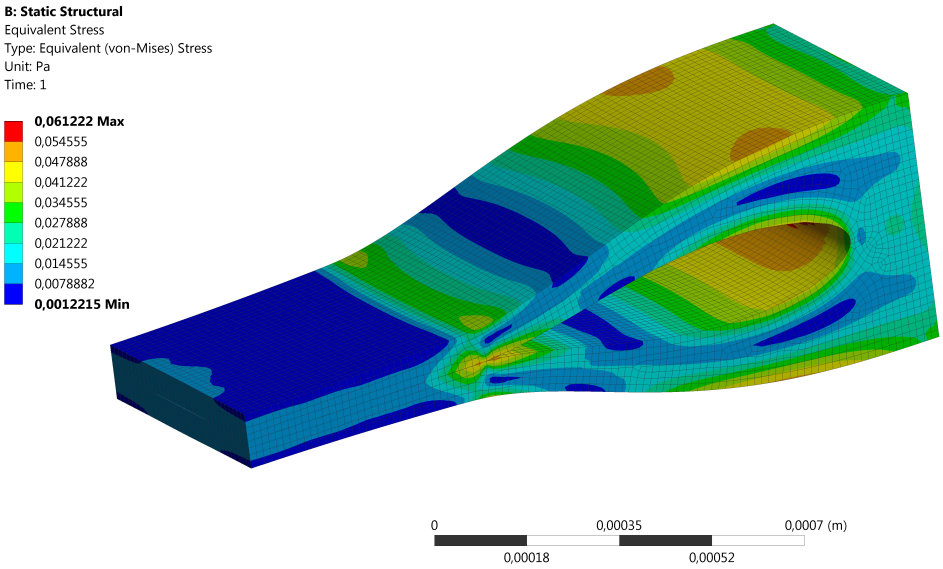


Figure 7.12: Ansys results for the 3D self-contact problem

Chapter 8

Summary and Conclusions

The main objective of this work was to develop and implement a multi-purpose penalty contact algorithm for the finite cell method (FCM). Among the requirements were the applicability to two- and three-dimensional problems involving large deformations. Moreover, the algorithm in question had to be compatible with the finite cell method developed at the Chair for Computation in Engineering of the Technische Universität München.

In order to fulfill these, it was necessary to first perform some general research on the FCM and the field of computational contact Mechanics. Chapter 2 introduced the finite element method before moving on to the FCM and more advanced concepts such as the application to large deformations. Chapter 4 was dedicated to two different global search algorithms, the pinball and spatial sorting. A comparison between the two showed a clear advantage in run time for the spatial sorting contact search due to its sorting algorithm of complexity $O(N \log(N))$. Another mandatory ingredient for the described contact algorithm is the local search, which in this case corresponds to a closest-point projection procedure. The techniques used for linear segments, planar triangles and quadrilaterals as well as a discussion about the extension to the analysis of arbitrarily discretized surfaces are the topic of chapter 5. Finally, the core of the present work, the regularization of contact constraints using the penalty method, is discussed in chapter 6. In this chapter, all the necessary tools for the derivation of the regulating equations as well as the linearization and discretization are presented, for both normal and tangential contact.

Some numerical examples are given in chapter 7. Results for the two-dimensional Hertz problem are shown for different setups, which differ from the polynomial order of the shape functions and the number of elements used. These were compared to the analytical solution proposed by Hertz in Hertz (1882). It can be clearly seen in the comparison that a higher polynomial order of the shape functions and a finer mesh lead to more accurate results. The solution obtained is very close to the analytical solution, especially in the middle of the contact region. A study has also been performed on the possibility to under-integrate the contact constraints on the slave boundary. This has shown important gains in computation time for the constraints without any notable effect on the computed maximum stresses. The second example, titled *Compression of a Foam*, demonstrated the possibility to simulate large deformation contact using the FCM and its relevance to real-world applications. The von Mises stress distribution obtained is coherent and corresponds to the expectations. The third example focuses on a very popular problem in the field of computational contact mechanics,

the ironing problem. A three-dimensional analysis was performed using a sphere with a FCM mesh and a high-order finite element mesh for the master surface. Lastly, a simple three-dimensional self-contact example, which reminds of the foam compression, is presented. The results have been compared with a similar problem solved using the commercial software Ansys. A good correlation between the two has been observed.

For further improvement, many topics could be addressed. The first step would be to use error estimation techniques to combine the algorithm developed in the course of the current work with adaptive techniques. This has the potential to drastically improve the solution, as shown in Franke (2011) for contact with high-order elements. Another subject that could prove interesting is the development of a more efficient local search technique for planar quadrilaterals. The current one, described in section 5.1, is a general closest-point procedure which is applicable to arbitrary surfaces. It uses a relatively expensive iteration process (in comparison to the other cases) and could certainly be improved for a simple case such as the planar quadrilateral. A topic that should be addressed is the extension of the algorithm to the analysis of arbitrarily described surfaces. This particular point would require some important changes in the code, as it currently expects a description of the boundary as a series of linear segments, such as what is obtained from the marching squares or marching cubes algorithms.

Directional Derivative

A *directional derivative* represents the rate of change of a function along a specified vector at a certain point. It is a generalization of the general concept of derivative. It is often written as $D\mathbf{R}(\mathbf{x}_k)[\mathbf{u}]$, which can be interpreted as the derivative of $\mathbf{R}(\mathbf{x}_k)$ in the direction of the vector $\mathbf{u}$ at the point $\mathbf{x}$. This concept is illustrated in figure 1. According to Bonet and Wood (2008), the result of $D\mathbf{R}(\mathbf{x}_k)[\mathbf{u}]$ is always be linear in $\mathbf{u}$. This procedure is often referenced to as *linearization*. The directional derivative can be defined as in Bonet and Wood (2008):

$$D\mathbf{R}(\mathbf{x}_k)[\mathbf{u}] = \frac{d}{d\epsilon}\Big|_{\epsilon=0} \mathbf{R}(\mathbf{x}_k + \epsilon\mathbf{u}) \tag{1}$$

A general Newton-Raphson algorithm can be formulated for systems of non-linear equations as follows:

$$D\mathbf{R}(\mathbf{x}_k)[\mathbf{u}] = -\mathbf{R}(\mathbf{x}_k) \tag{2}$$

The first term of equation 2, which contains the directional derivatives of the equations in the direction of $\mathbf{u}$, is called *tangent matrix*. The right-hand side is referred to as the *residual*.

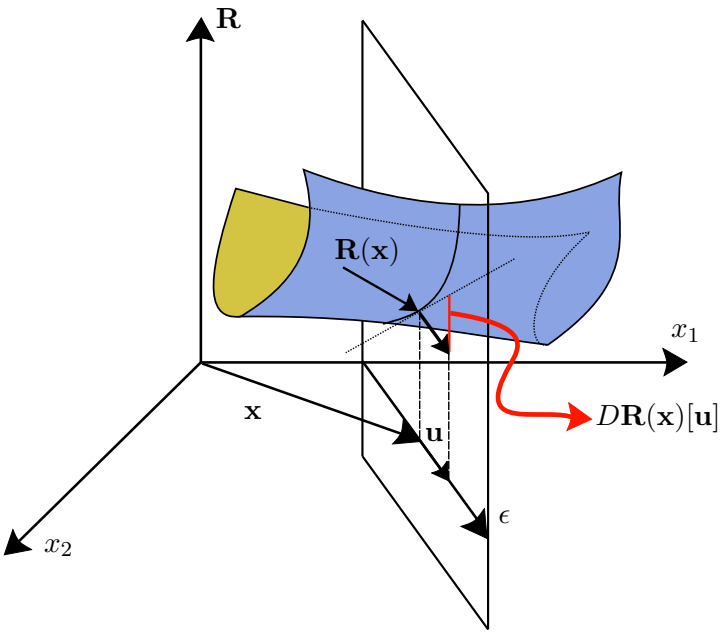


Figure 1: Illustration of the directional derivative with 2 variables

List of Figures

List of Tables

Bibliography

Bathe, K.-J., 1996. Finite Element Procedures. Prentice Hall.

Bog, T., Zander, N., Kollmannsberger, S., Rank, E., 2015. Normal contact with high order finite elements and a fictitious contact material. Computers & Mathematics with Applications 70 (7), 1370 – 1390.
URL http://www.sciencedirect.com/science/article/pii/S0898122115002011

Bonet, J., Wood, R. D., 2008. Nonlinear continuum mechanics for finite element analysis. Cambridge University Press, New York.

Chernyaev, E., 1995. Marching cubes 33: Construction of topologically correct isosurfaces. In: CERN Report, CN/95-17.

Cottrell, J. A., Hughes, T. J. R., Bazilevs, Y., 2009. Isogeometric Analysis: Toward Integration of CAD and FEA, 1st Edition. Wiley Publishing.

Dias, A. P. C., Serpa, A. L., Bittencourt, M. L., 2015. High-order mortar-based element applied to nonlinear analysis of structural contact mechanics. Computer methods in applied mechanics and engineering 294, 19–55.

Düster, A., 2008. High-Order FEM. Lectures notes, Chair for Computation in Engineering, Technische Universität München.

Düster, A., Parvizian, J., Yang, Z., Rank, E., 2008. The finite cell method for three-dimensional problems of solid mechanics. Computer Methods in Applied Mechanics and Engineering 197 (45-48), 3768 – 3782.

Felippa, C. A., 2015. Nonlinear finite element methods. Lectures notes, Department of Aerospace Engineering Sciences, University of Colorado.

Fortin, A., 2008. Analyse numérique pour ingénieurs. Presses internationales Polytechnique.

Franke, D., Düster, A., Nübel, V., Rank, E., 2010. A comparison of the h-, p-, hp-, and rp-version of the FEM for the solution of the 2D Hertzian contact problem. Computational Mechanics 45 (5), 513–522.
URL http://dx.doi.org/10.1007/s00466-009-0464-6

Franke, D. C., 2011. Investigation of mechanical contact problems with high-order finite element methods. Ph. D. thesis, Technische Universität München.

Heidrich, W., 2005. Computing the barycentric coordinates of a projected point. Journal of Graphics, GPU, and Game Tools 10 (3), 9–12.
URL http://dx.doi.org/10.1080/2151237X.2005.10129200

Hertz, H., 1882. Über die Berührung fester elastischer Körper. Journal für die reine und angewandte Mathematik 92, 156–171.

Hughes, T. J. R., 2000. The Finite Element Method: Linear Static and Dynamic Finite Element Analysis. Dover Civil and Mechanical Engineering. Dover Publications.

Kim, N.-H., 2015. Introduction to Nonlinear Finite Element Analysis. Springer Science+Business Media New York.

Kim, T., Dolbow, J., Laursen, T., 2007. A mortared finite element method for frictional contact on arbitrary interfaces. Computational Mechanics 39 (3), 223–235.
URL http://dx.doi.org/10.1007/s00466-005-0019-4

Konyukhov, A., 2011. Geometrically exact theory for contact interactions. Habilitation thesis, Karlsruhe Institut für Technologie.

Konyukhov, A., Izi, R., 2015. Introduction to Computational Contact Mechanics: a geometrical approach. John Wiley & Sons Ltd.

Konyukhov, A., Lorenz, C., Schweizerhof, K., 2015. Various contact approaches for the finite cell method. Computational Mechanics 56 (2), 331–351.
URL http://dx.doi.org/10.1007/s00466-015-1174-x

Konyukhov, A., Schweizerhof, K., 2004. Contact formulation via a velocity description allowing efficiency improvements in frictionless contact analysis. Computational Mechanics 33 (3), 165–173.
URL http://dx.doi.org/10.1007/s00466-003-0515-3

Konyukhov, A., Schweizerhof, K., 2005a. Covariant description for frictional contact problems. Computational Mechanics 35, 190–213.

Konyukhov, A., Schweizerhof, K., 2005b. A special focus on 2D formulations for contact problems using a covariant description. International Journal for Numerical Methods in Engineering 66, 1432–1465.

Konyukhov, A., Schweizerhof, K., 2008a. Incorporation of contact for high-order finite elements in covariant form. Computer Methods in applied mechanics and engineering 198, 1213–1223.

Konyukhov, A., Schweizerhof, K., 2008b. On the solvability of closest point projection procedures in contact analysis: Analysis and solution strategy for surfaces of arbitrary geometry. Computer Methods in Applied Mechanics and Engineering 197 (33-40), 3045 – 3056.

Lorensen, W. E., Cline, H. E., Aug. 1987. Marching cubes: A high resolution 3D surface construction algorithm. SIGGRAPH Comput. Graph. 21 (4), 163–169.
URL http://doi.acm.org/10.1145/37402.37422

Lu, J., 2011. Isogeometric contact analysis: Geometric basis and formulation for frictionless contact. Computer Methods in applied mechanics and engineering 200, 726–741.

Luenberger, D. G., 1984. Linear and Nonlinear Programming, 1st Edition. Addison-Wesley, Reading.

Matzen, M., Cichosz, T., Bischoff, M., 2013. A point to segment contact formulation for isogeometric, NURBS based finite elements. Computer Methods in Applied Mechanics and Engineering 255, 27 – 39.
URL http://www.sciencedirect.com/science/article/pii/S0045782512003490

Nitsche, J., 1971. Über ein Variationsprinzip zur Lösung von Dirichlet-Problemen bei Verwendung von Teilräumen, die keinen Randbedingungen unterworfen sind. Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg 36 (1), 9–15.
URL http://dx.doi.org/10.1007/BF02995904

O'Connor, R., 1996. A distributed discrete element modelling environment - algorithms, implementation and applications. Ph. D. thesis, Massachusetts Institute of Technology.

Papadopoulos, P., Jones, R. E., Solberg, J. M., 1995. A novel finite element formulation for frictionless contact problems. International Journal for Numerical Methods in Engineering 38, 2603–2617.

Parvizian, J., Düster, A., Rank, E., 2007. Finite cell method. Computational Mechanics 41 (1), 121–133.
URL http://dx.doi.org/10.1007/s00466-007-0173-y

Persson, B. N. J., 2000. Sliding Friction - Physical Principles and Applications. Springer-Verlag Berlin Heidelberg.

Schillinger, D., Ruess, M., 2014. The finite cell method: A review in the context of higher-order structural analysis of CAD and image-based geometric models. Archives of Computational Methods in Engineering, 1–65.
URL http://dx.doi.org/10.1007/s11831-014-9115-y

Schillinger, D., Ruess, M., Zander, N., Bazilevs, Y., Düster, A., Rank, E., 2012. Small and large deformation analysis with the p- and b-spline versions of the finite cell method. Computational Mechanics 50 (4), 445–478.
URL http://dx.doi.org/10.1007/s00466-012-0684-z

Schweizerhof, K., Konyukhov, A., 2005. Covariant description for frictional contact problems. Computational Mechanics 35 (3), 190–213.
URL http://dx.doi.org/10.1007/s00466-004-0616-7

Szabó, B., Düster, A., Rank, E., 2004. The p-version of the finite element method. Encyclopedia of computational mechanics, 119–139.

Vos, P., van Loon, R., Sherwin, S., 2008. A comparison of fictitious domain methods appropriate for spectral/hp element discretisations. Computer Methods in Applied Mechanics and Engineering 197 (25-28), 2275–2289.
URL http://www.sciencedirect.com/science/article/pii/S0045782507004732

Williams, J. R., O'Connor, R., 1995. A linear complexity intersection algorithm for discrete element simulation of arbitrary geometries. Engineering Computations 12, 185–201.

Wriggers, P., 2006. Computational Contact Mechanics. Springer-Verlag Berlin Heidelberg.

Wriggers, P., Simo, J. C., 1985. A note on tangent stiffness for fully nonlinear contact problems. Communications in applied numerical methods 1, 199–203.

Yang, D., 2001. C++ and object-oriented numeric computing for scientists and engineers. Springer-Verlag New York, Inc.

Yastrebov, V. A., 2011. Computational contact mechanics: geometry, detection and numerical techniques. Ph. D. thesis, MINES ParisTech.

Zander, N., Bog, T., Elhaddad, M., Espinoza, R., Hu, H., Joly, A., Wu, C., Zerbe, P., Düster, A., Kollmannsberger, S., Parvizian, J., Ruess, M., Schillinger, D., Rank, E., 2014. Fcmlab: A finite cell research toolbox for MATLAB. Advances in Engineering Software 74, 49 – 63.
URL http://www.sciencedirect.com/science/article/pii/S0965997814000684

Zander, N., Bog, T., Kollmannsberger, S., Schillinger, D., Rank, E., 2015. Multi-level hp-adaptivity: high-order mesh adaptivity without the difficulties of constraining hanging nodes. Computational Mechanics 55 (3), 499–517.
URL http://dx.doi.org/10.1007/s00466-014-1118-x

Zienkiewicz, O. C., Taylor, R. L., 1977. The finite element method. Vol. 3. McGraw-hill London.

Zienkiewicz, O. C., Taylor, R. L., 2005. The finite element method for solid and structural mechanics. Butterworth-heinemann.