Department of Construction Sciences

Solid Mechanics

ISRN LUTFD2/TFHF–07/5128-SE(1-60)

AN IMPLEMENTATION OF MESH FREE METHODS FOR MECHANICAL PROBLEMS AT LARGE STRAINS

Master Thesis by

VIKTOR PETERSSON

SUPERVISOR

PH. D. PAUL HÅKANSSON

EXAMINER

PROF. MATTI RISTINMAA

Preface

This master thesis are the last part of my degree in Master of Science in Engineering Mathematics. The project has been performed under the period from February to August 2007 at the division of Solid Mechanics.

First I like to thank my examiner Prof. Matti Ristinmaa who introduced me to this exciting research subject and gave me the opportunity to write this master thesis. Then I would like to give my supervisor Ph. D. Paul Håkansson an extra thanks for all his time and effort he spend on guiding me under this project. I really appreciate that he always took time and helped me through my endless questions. I had never managed to do this without his help. Also a big thanks to the rest of the staff at division of Solid Mechanics that helped to make this a enjoyable and unforgettable six months of my life.

Finally I would like to thank family, friends and my beloved Sara for their interest and constant encouragement.

Lund, August 2007

Viktor Petersson

Abstract

Simulation of different engineering applications has for many years been a large industry. This is done by solving partial differential equation with initial values and boundary conditions. The Finite Element Method (FEM) has been the standard tool for this kind of calculations. But under the last fifteen years a new mesh free method has been under extensive research. In mesh free methods there is no element that combine the nodes. Without this restriction of connectivity between the nodes, mesh free methods have some advantages in special applications.

It is well known that the mesh less methods are more time consuming than the FEM. So for now mesh free methods is not any threat to the FEM in standard simulations. But in special applications mesh free methods have advantages that FEM do not have. Examples of these applications are large deformation and discontinuous problems, for example crack growth or interface problem. Examples of interface problems are solid-solid (two different materials) or solid-fluid. In large deformation analysis with FEM, the element can suffer from large distortion and remeshing is necessary. This remeshing is time consuming and a projection of the field variables have to be made, which can introduce errors in the calculations. In discontinuous problem you can for example describe the crack without adding nodes, and again no remeshing is needed.

This master thesis give an introduction to this large research subject. After literature study, two methods was chosen and implemented for linear static. The Element Free Galerkin (EFG) method and the Reproducing Kernel Particle Method (RKPM), which are the two most widely used methods. The both methods have a quite different approach how to construct shape functions. A comparison is made between the two methods concerning accuracy, effectiveness and implementation difficulties. Despite the two different approaches, they are surprisingly similar in performance and accuracy. Because of the similar results, only RKPM is further developed to manage large deformation analysis.

The step to large deformation analysis is done in the following parts. First a material non-linear model for plasticity with linear kinematic hardening is implemented. After that a geometric non-linear model for hyperelasticity, then finally a elasto-plastic model for large deformation with a non-linear isotropic hardening. The elasto-plastic model follows from a multiplicative split of the deformation gradient, but no further theory is presented in this report.

For all cases numerical examples are discussed and compared against the commercial FEM program ABAQUS. For the two first cases there is excellent agreement. But for the last example, necking of a bar, the solution is not trivial and even ABAQUS give different results depending which elements that are used. Also RKPM give different results when changing parameters, so a more detailed investigation and some experimental data would be needed to give any conclusion about the results. In general the mesh free program work well and give fine results, but there are many loose ends that would needed some more attention.

Contents

Chapter 1

Introduction to Mesh Free Methods

As for now the finite element method has been a powerful tool for solving partial differential equations. It has successfully been applied for a large number of engineering applications, for example solid mechanics, structure mechanics, electro magnetism, geo mechanics, bio mechanics and so on. But for the last fifteen years a new mesh free method has been subject to extensive research.

Compared to the finite element method, the mesh free methods do not have any connectivity between the nodes (the elements in FEM). This give some advantages in applications as large deformations. Large deformation analysis in FEM give rise to remeshing because the elements are highly distorted. This remeshing is computational heavy and accuracy is lost because a projection has to be made between the two meshes. In mesh free methods this problem does not occur. Other applications were mesh free methods has been applied are for example crack simulation and interface problem. This master thesis is focused on investigating the methods and apply it to large deformation problem.

1.1 Overview of mesh free methods

In this section a general procedure for how the mesh free approximation are constructed and some problem that occur is discussed. This section is just to introduce the reader to some important concepts and compare the difference against FEM. The notation in Table 1.1 is consistent through the report.

1.1.1 Weak form and corresponding mesh free form

As for FEM the goal is to solve a partial differential equation with initial values and boundary conditions. The weak form is constructed exactly as in FEM. There is no unified way how to choose the test function Ψ_i in mesh free methods. In FEM it is chosen as Galerkin, i.e. $\Psi_i = \phi_i$, and that is most common in mesh free methods too. But also point collocation, $\Psi_i = \delta(x - x_i)$ is widely used. A benefit with point

symbol	explanation
u	unknown function
$\mathbf{x}$	space coordinate
$\mathbf{x}_i$	position of a node
u_i	nodal value
ϕ	shape function
N	total number of nodes
m	number of basis coefficients
n	number of nodes inside support domain
Ω	problem domain
Ω_I	support domain to node I

Table 1.1: Variables and their explanation.

collocation is that you do not need to integrate over the problem domain Ω, it solves the strong form of the problem. But it is well known to have problem with accuracy. Then there are a number of other ways to choose the test function, for a summarize see for example Fries and Matthies (2004). A slightly more different way is to use a local weak form, instead of a global. This also have the benefit that you do not need to integrate over Ω. This idea was originated by Atluri and Zhu (1998).

1.1.2 Shape functions and support domain

After choosen the testfunction, the function u is approximated. Two of the most used methods to construct shape functions are the Moving least square (MLS) method and the Reproducing Kernel particle method (RKPM). More details about these methods will follow later on in the report, but roughly MLS method comes from data fitting theory and RKPM from theory of wavelets. There are several other methods, but they are not discussed in this report. Common for these methods is that they end up with an approximation similar to FEM

$$u(\mathbf{x}) \approx \sum_{i=1}^{n} \phi_i u_i. \tag{1.1}$$

Because there is no connectivity between the nodes, you have to decide which nodes $\mathbf{x}_i$ should influence on the approximation for a point $\mathbf{x}$. It is not computationally possible to use all the nodes in Ω, therefore we introduce a very important expression called support domain. Nodes inside the support domain contribute to the approximation. Often you have circular or rectangular support domains, and they are often the same for all points in the domain, but it is not necessary. For example of support domains see Figure 1.1.

In practice to realize this support domain, the shape functions are multiplied with a weight function. This weight function are non zero inside the support domain, and

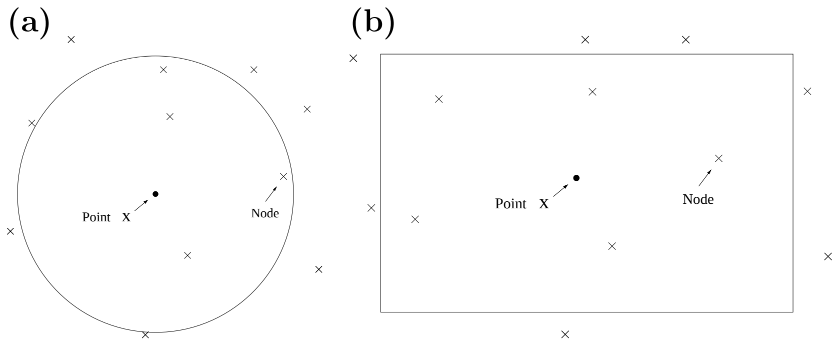

Figure 1.1: Example of support domain: a) circular b) rectangular

have compact support, i.e. zero outside the support domain. Examples of weight functions follows later on in the report.

1.1.3 Invoke essential boundary conditions.

Most of these methods to construct shape functions do not fulfill the Kronecker delta property, i.e. if you construct a shape function for a node it should be one for the current node and zero for all other. Or in more mathematical terms

$$\phi_i(\mathbf{x}_j) = \delta_{ij}. \tag{1.2}$$

The lack of this property give rise to some problem. When you solve the system of equations you do not get real values of the field variables, just fictive values. To get the real value you have to perform a search for all nodes in the support domain and use the shape functions. A bigger problem are the essential boundary. In FEM you have Kronecker delta property, so when you have a prescribed displacement at a node you can just put in this value at the equation system. This is not the case for mesh less methods, something else has to be done.

There are in general three methods to deal with this problem, and of course another hundred if you start looking in the literature.

- **Lagrange Multiplier** is accurate and easy to implement. But it has large numerical costs because you get a larger equationsystem and you loose a banded stiffness matrix.
- **Penalty method** is also easy to implement and the numerical costs is small. But it is not very accurate and therefore not widely used.
- **Update shape functions** so they posses Kronecker delta property. This leads to easy enforcement of boundary condition, but the implementation of shape functions get more tricky. This is of course the best alternative if you get a easy implementation. See for examples articles by Most and Bucher (2007) and Rossi and Alves (2007).

Lagrange multiplier is the most widely used of these three, but as more sofisticated shape functions with Kronecker delta property come they will replace Lagrange multiplier. For details and a more extensive comparison between Lagrange multiplier and penalty methods, see article by Gavete *et al.* (2000).

1.1.4 Implementation issues

If you compare the general structure of a mesh free and a FEM program, both starts with a discretization of Ω. Depending on which test function is used, a numerical integration has to be performed. For a Galerkin formulation an integral over Ω is necessary. The usual integration scheme is Gauss- or Direct Nodal integration. In Gauss integration Ω is divided in cells, that are completely independent of the nodes, which contains Gauss points. In direct nodal integration the integral is evaluated only at the nodes. Benefits with nodal integration is that you do not need to generate integration points and it is faster than full integration. But similar to point collocation methods there are some stability and accuracy problem.

Also Gauss integration have problems with accuracy, because the support and integration cells do not coincide. In FEM you integrate over the element which is a smooth for example second order polynomial, so the numerical integration gives the exact result. In mesh free methods the shape function is much more complex. In practice it is easy to get sufficiently accurate results, but the number of integration points has to be larger than regular FEM.

After you have created nodes and integration points, you start to compute the system matrices. In FEM this is done by a loop over all elements, and local matrices is calculated and assembled to right position. In mesh less methods the loop is over all integration points. The number of nodes inside the support domain determine the size of the local matrix which is then assembled to the global. Then you solve the system of equations and get you fictive nodal values.

Then some post processing has to be done to get the real displacement at a point. In FEM this is not necessary, but in FEM you get a discontinuous first derivate of the function. So if you want to know the strain for the entire body, you need to do some interpolation. This is not necessary for the mesh less methods. There you get approximation that have continuous derivatives.

It is well known that the mesh less methods are more time consuming then FEM. That depends on the more complex shape functions and the need for many integration points to get accurate results.

1.2 Examples of mesh free methods

Over the last years a number of different mesh free methods have been developed. There is no possibility to mention them all in this section, for a more extensive review and classification of all mesh free methods see the excellent article by Fries and Matthies (2004). As explained in the last section there is a number of choice you have to make, for example construction of shape function, test function, integration method and so on. Each of these combinations have different names, but there are also some methods that origin from a completely other point of view. In the following list some of the most well known methods are described

- **Element Free Galerkin** (EFG) was developed by Belytschko *et al.* (1994). The shape functions are constructed with MLS approximation, and the test function is chosen as the shape function. Boundary condition are enforced by Lagrange multiplier and in general a lot of gauss integration points is needed to get accurate results.

- **Smooth Particle Hydrodynamics** (SPH) was introduced by Lucy (1977) and further developed by Monaghan (1982). It is the most simple method, partly because it is a point collocation method and also because the shape functions are very simple with no special cases at the boundary. It has some problems with both stability and accuracy, but many corrections have been made to improve the method, for example better integration scheme and correction term in the shape functions.

- **Reproducing Kernel Particle Method** (RKPM) was created by Liu *et al.* (1995). It is a particle method, but instead of point collocation it uses a Galerkin formulation. Also the shape function have a correction term to improve the accuracy at the boundary.

- **Mesh less Local Petrov-Galerkin** (MLPG) was originated by Atluri and Zhu (1998). Instead of a global weak form, it have an local weak form. Therefore no integration over the domain is necessary, so no background mesh is needed as for example EFG and RKPM. MLPG can have different shape functions and test functions and is then named as MLPG1, MLPG2 and so on. Common for all are the local weak form.

- **Natural Element Method** (NEM) was developed by Sukumar *et al.* (1998). Also NEM solves a Galerkin formulation of the problem. But here the shape functions are constructed in a different fashion. The domain is divided in Voroni cells and the shape function value for a point $\mathbf{x}$ with respect to a node $\mathbf{x}_i$ is the ratio between area of $\mathbf{x}$ overlap to $\mathbf{x}_i$ and the total area of $\mathbf{x}$, i.e $\phi_i(\mathbf{x}) = \frac{A_i(\mathbf{x})}{A(\mathbf{x})}$. So the NEM fulfills the Kronecker delta property and therefore it is straightforward to implement essential boundary conditions.

These are some of the uncountable methods that exist in the literature. The question arise, which method was first, and there is no good answer to that question. But if some methods should be mentioned, Diffuse element method (Nayroles *et al.* (1992)), which was a forerunner to EFG, and SPH was one of the first.

Chapter 2

Galerkin mesh free formulation with Lagrange multiplier

2.1 Introduction

As mentioned before there are a number of different mesh free methods. Two of the most used methods are Element Free Galerkin and Reproducing Kernel Particle Method. They are very different how the shape functions is constructed, therefore they are implemented in this master thesis. Common for both methods is that they use Galerkin formulation, which makes the methods stable. In this master thesis both use Lagrange multiplier to enforce essential boundary condition. The weak form and the corresponding mesh free formulation is derived that are valid for both methods.

2.2 Weak form for Solid Mechanics

The partial differential equation that controls solid mechanics can for 2-D be stated as

$$\mathbf{R}^T\boldsymbol{\sigma} + \mathbf{b} = 0, \tag{2.1}$$

for derivation see Ottosen and Petersson (1992). This equation apply for all points in the problem domain Ω. For the essential boundary, S_u, we have a prescribed displacement

$$\mathbf{u} = \overline{\mathbf{u}}.$$

For the natural boundary, S_t, we have a prescribed force given by the traction vector

$$\overline{\mathbf{t}} = \mathbf{Sn}$$

where $\mathbf{n}$ is the normal vector to the boundary and $\mathbf{S}$ is the stress tensor. The matrices in the formulation is given by

$$\begin{aligned}
\mathbf{R} &= \begin{bmatrix} \frac{\partial}{\partial x} & 0 \\ 0 & \frac{\partial}{\partial y} \\ \frac{\partial}{\partial y} & \frac{\partial}{\partial x} \end{bmatrix} && \text{Operator working on the stress} \\
\boldsymbol{\sigma} &= \begin{bmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \sigma_{xy} \end{bmatrix} && \text{Stressmatrix} \\
\mathbf{b} &= \begin{bmatrix} b_x \\ b_y \end{bmatrix} && \text{Forces working on the body} \qquad . \\
\mathbf{u} &= \begin{bmatrix} u_x \\ u_y \end{bmatrix} && \text{Displacement vector for a point} \\
\mathbf{n} &= \begin{bmatrix} n_x \\ n_y \end{bmatrix} && \text{Normal vector to a boundary point} \\
\mathbf{S} &= \begin{bmatrix} \sigma_{xx} & \sigma_{xy} \\ \sigma_{yx} & \sigma_{yy} \end{bmatrix} && \text{Stress tensor}
\end{aligned}$$

This is the strong form of the problem. In general we can not solve this equation analytically. This is why we need numerical methods like the finite element method. In order to solve the problem we need to lower the regularity of the function $\mathbf{u}$ we seek. There are several methods for doing this, the one used in this report is a variational principle. We construct the lagrangian function L that in solid mechanics is calculated according to

$$L = T - \Pi + W \tag{2.2}$$

where T is the kinetic energy, Π is the elastic energy and W is the work done by the external forces. The components of the function L is calculated as

$$T = \frac{1}{2}\int_{\Omega} \rho \dot{\mathbf{u}}^T \dot{\mathbf{u}} \, d\Omega \tag{2.3}$$

$$\Pi = \frac{1}{2}\int_{\Omega} \boldsymbol{\varepsilon}^T \boldsymbol{\sigma} \, d\Omega \tag{2.4}$$

$$W = \int_{\Omega} \mathbf{u}^T \mathbf{b} \, d\Omega + \int_{S_t} \mathbf{u}^T \mathbf{t} \, dS_t. \tag{2.5}$$

In EFG and RKPM the shape functions do not fulfill the Kroneckers delta property. So in order to invoke essential boundary, we have to use Lagrange multiplier. This will lead to a modified Lagrange function

$$\tilde{L} = L + \int_{S_u} \boldsymbol{\lambda}^T (\mathbf{u} - \overline{\mathbf{u}}) \, dS_u. \tag{2.6}$$

The Lagrange multiplier ($\boldsymbol{\lambda}$) can be interpreted as the reaction forces needed to fulfill the displacement conditions at the boundary.

Hamilton's theorem states that the variation of Lagrange function is equal to zero,

$$\delta \tilde{L} = 0. \tag{2.7}$$

By using expression 2.6 we get

$$\delta(\frac{1}{2}\int_\Omega \rho\dot{\mathbf{u}}^T\dot{\mathbf{u}}\ d\Omega - \frac{1}{2}\int_\Omega \boldsymbol{\varepsilon}^T\boldsymbol{\sigma}\ d\Omega + \int_\Omega \mathbf{u}^T\mathbf{b}\ d\Omega + \\ + \int_{S_t}\mathbf{u}^T\mathbf{t}\ dS_t + \int_{S_u}\boldsymbol{\lambda}^T(\mathbf{u}-\overline{\mathbf{u}})\ dS_u) = 0. \tag{2.8}$$

By simplify the elastic term of the Lagrange equation

$$\delta(\boldsymbol{\varepsilon}^T\boldsymbol{\sigma}) = \delta\boldsymbol{\varepsilon}^T\boldsymbol{\sigma} + \boldsymbol{\varepsilon}^T\delta\boldsymbol{\sigma} \tag{2.9}$$

where

$$\boldsymbol{\varepsilon}^T\delta\boldsymbol{\sigma} = (\boldsymbol{\varepsilon}^T\delta\boldsymbol{\sigma})^T = \delta\boldsymbol{\sigma}^T\boldsymbol{\varepsilon} = \delta(\mathbf{c}\boldsymbol{\varepsilon})^T\boldsymbol{\varepsilon} = \delta\boldsymbol{\varepsilon}^T\mathbf{c}^T\boldsymbol{\varepsilon} = \delta\boldsymbol{\varepsilon}^T\mathbf{c}\boldsymbol{\varepsilon} = \delta\boldsymbol{\varepsilon}^T\boldsymbol{\sigma}. \tag{2.10}$$

Here we assume that the constitutive matrix $\mathbf{c}$ is symmetric. If we use this result and static consideration, i.e. the kinetic term of the lagranian vanish, equation 2.8 reduces to

$$-\int_\Omega \delta\boldsymbol{\varepsilon}^T\boldsymbol{\sigma}\ d\Omega + \int_\Omega \delta\mathbf{u}^T\mathbf{b}\ d\Omega + \int_{S_t}\delta\mathbf{u}^T\mathbf{t}\ dS_t + \int_{S_u}\delta\boldsymbol{\lambda}^T(\mathbf{u}-\overline{\mathbf{u}})\ dS_u + \int_{S_u}\boldsymbol{\lambda}^T\delta\mathbf{u}\ dS_u = 0. \tag{2.11}$$

The last term is a scalar, therefore nothing will change if we take the transpose

$$\int_{S_u}\boldsymbol{\lambda}^T\delta\mathbf{u}\ dS_u = \int_{S_u}(\boldsymbol{\lambda}^T\delta\mathbf{u})^T\ dS_u = \int_{S_u}\delta\mathbf{u}^T\boldsymbol{\lambda}\ dS_u. \tag{2.12}$$

So equation 2.11 becomes

$$-\int_\Omega \delta\boldsymbol{\varepsilon}^T\boldsymbol{\sigma}\ d\Omega + \int_\Omega \delta\mathbf{u}^T\mathbf{b}\ d\Omega + \int_{S_t}\delta\mathbf{u}^T\mathbf{t}\ dS_t + \int_{S_u}\delta\boldsymbol{\lambda}^T(\mathbf{u}-\overline{\mathbf{u}})\ dS_u + \int_{S_u}\delta\mathbf{u}^T\boldsymbol{\lambda}\ dS_u = 0. \tag{2.13}$$

This is the weak form for a solid mechanic problem. At this point we have not introduced any kind of approximations, this is a law of nature modified a little bit. In order to discretize the domain we have to invoke some kind of approximation.

2.3 Introducing approximation

With the approximation given by

$$u(\mathbf{x}) \approx \sum_{i=1}^{n}\phi_i u_i = \boldsymbol{\phi}\mathbf{U} \tag{2.14}$$

we can now approach the final formulation for a linear static problem in solid mechanics. But first the unknown Lagrange multiplier also, like the displacement, has to be discretized on the essential boundary. Like the displacement it is approximated as sum of nodal values multiplied with a shape function, $\boldsymbol{\lambda} = \mathbf{N}\boldsymbol{\Lambda}$. A common use as $\mathbf{N}$ is, like in FEM, the simplest linear Lagrange shape functions, that only depends on the nodes to the left and right of the current node. Inserted the equation 2.13 results in

$$
\begin{aligned}
-\int_{\Omega} \delta\boldsymbol{\varepsilon}^T\boldsymbol{\sigma}\, d\Omega + \int_{\Omega} \delta\mathbf{U}^T\boldsymbol{\phi}^T\mathbf{b}\, d\Omega + \int_{S_t} \delta\mathbf{U}^T\boldsymbol{\phi}^T\mathbf{t}\, dS_t + \\
+\int_{S_u} \delta\boldsymbol{\Lambda}^T\mathbf{N}^T\boldsymbol{\phi}\mathbf{U} - \int_{S_u} \delta\boldsymbol{\Lambda}^T\mathbf{N}^T\overline{\mathbf{u}}\, dS_u + \int_{S_u} \delta\mathbf{U}^T\boldsymbol{\phi}^T\mathbf{N}\boldsymbol{\Lambda}\, dS_u = 0.
\end{aligned} \tag{2.15}
$$

For now no assumption has been made concerning constitutive relation. If small strains are considered the following relation holds between the displacement and the strains, $\boldsymbol{\varepsilon} = \mathbf{Ru} = \mathbf{R}\boldsymbol{\phi}\mathbf{U} = \mathbf{BU}$. And for linear elasticity the stress is obtained by $\boldsymbol{\sigma} = \mathbf{c}\boldsymbol{\varepsilon}$. The nodal values vectors $\mathbf{U}$ and $\boldsymbol{\Lambda}$ are independent of $\mathbf{x}$, so they can be detached from the integral

$$
\begin{aligned}
-\delta\mathbf{U}^T \underbrace{\int_{\Omega} \mathbf{B}^T\mathbf{c}\mathbf{B}\, d\Omega}_{\mathbf{K}} \mathbf{U} + \delta\mathbf{U}^T \underbrace{\left(\int_{\Omega} \boldsymbol{\phi}^T\mathbf{b}\, d\Omega + \int_{S_t} \boldsymbol{\phi}^T\mathbf{t}\, dS_t\right)}_{\mathbf{F}} \\
-\delta\boldsymbol{\Lambda}^T \underbrace{\int_{S_u} -\mathbf{N}^T\boldsymbol{\phi}\, dS_u}_{\mathbf{G}^T} \mathbf{U} + \delta\boldsymbol{\Lambda}^T \underbrace{\int_{S_u} -\mathbf{N}^T\overline{\mathbf{u}}\, dS_u}_{\mathbf{q}} -\delta\mathbf{U}^T \underbrace{\int_{S_u} -\boldsymbol{\phi}^T\mathbf{N}\, dS_u}_{\mathbf{G}} \boldsymbol{\Lambda} = 0.
\end{aligned} \tag{2.16}
$$

This can be written as

$$
\delta\mathbf{U}^T(\mathbf{KU} - \mathbf{F} + \mathbf{G}\Lambda) + \delta\boldsymbol{\Lambda}^T(\mathbf{G}^T\mathbf{U} - \mathbf{q}) = 0. \tag{2.17}
$$

Because both variations are independent, and not always equals to zero, the terms they are multiplied with have to be zero. This gives the following equation system

$$
\begin{bmatrix} \mathbf{K} & \mathbf{G} \\ \mathbf{G}^T & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{U} \\ \boldsymbol{\Lambda} \end{bmatrix} = \begin{bmatrix} \mathbf{F} \\ \mathbf{q} \end{bmatrix}. \tag{2.18}
$$

Now the unknowns nodal parameters can be solved by inverting the left matrix. This is the mesh free formulation for linear static.

Chapter 3

Element Free Galerkin Method

3.1 Introduction

In Element Free Galerkin (EFG) we use the moving least square (MLS) method for constructing the shape functions. Moving least square method was first proposed by Lancaster and Salkauskas (1981), as an interpolation method. It was used in element free methods by Belytschko *et al.* (1994), with use of Lagrange multiplier to invoke essential boundary.

The MLS approximation of the displacement field basically looks like this. We approximate the displacement in a point by coefficients to a polynomial basis **p**. Then we minimize a weighted error function J in a least square sense to get our shape functions.

3.2 Deriving shape functions

We approximate the displacement field by a discrete sum

$$u(\mathbf{x}) \approx \hat{u}(\mathbf{x}) = \sum_{i=1}^{m} p_i(\mathbf{x}) a_i(\mathbf{x}) = \mathbf{p}^T(\mathbf{x})\mathbf{a}(\mathbf{x}) \tag{3.1}$$

where **p** is a basis. For example a linear basis in two dimensions becomes $\mathbf{p}^T = [1 \text{ x y}]$. In order to determine the unknown coefficients **a**, a functional J is constructed. It sum up the weighted quadratic error for all nodes inside the support domain as

$$J = \sum_{i=1}^{n} W(\mathbf{x} - \mathbf{x}_i)(\hat{u}_i - u_i)^2 = \sum_{i=1}^{n} W(\mathbf{x} - \mathbf{x}_i)(\mathbf{p}^T(\mathbf{x}_i)\mathbf{a}(\mathbf{x}) - u_i)^2 \tag{3.2}$$

where W is the weight function. Then we want to minimize this functional, so we differentiate with respect to the unknown vector **a**, containing the coefficient

$$\frac{\partial J}{\partial \mathbf{a}} = \mathbf{0}. \tag{3.3}$$

By inserting the expression for J, the equation ends up with

$$\begin{aligned}
\frac{\partial J}{\partial \mathbf{a}} &= \sum_{i=1}^{n} W(\mathbf{x}-\mathbf{x}_i)\frac{\partial(\mathbf{p}^T(\mathbf{x}_i)\mathbf{a}(\mathbf{x})-u_i)^2}{\partial \mathbf{a}} && (3.4)\\
&= \sum_{i=1}^{n} W(\mathbf{x}-\mathbf{x}_i)2(\mathbf{p}^T(\mathbf{x}_i)\mathbf{a}(\mathbf{x})-u_i)\mathbf{p}(\mathbf{x}_i) = 0 && (3.5)\\
\Leftrightarrow &\sum_{i=1}^{n} W(\mathbf{x}-\mathbf{x}_i)\mathbf{p}(\mathbf{x}_i)\mathbf{p}^T(\mathbf{x}_i)\mathbf{a}(\mathbf{x}) = \sum_{i=1}^{n} W(\mathbf{x}-\mathbf{x}_i)\mathbf{p}(\mathbf{x}_i)u_i. && (3.6)
\end{aligned}$$

This can be written in a compact matrix form as

$$\mathbf{A}(\mathbf{x})\mathbf{a}(\mathbf{x}) = \mathbf{B}(\mathbf{x})\mathbf{U}(\mathbf{x}) \qquad (3.7)$$

where the matrices are given by

$$\begin{aligned}
\mathbf{A}(\mathbf{x}) &= \sum_{i=1}^{n} W(\mathbf{x}-\mathbf{x}_i)\mathbf{p}(\mathbf{x}_i)\mathbf{p}^T(\mathbf{x}_i) \in \mathbf{M}(m\times m) && (3.8)\\
\mathbf{B}(\mathbf{x}) &= \begin{bmatrix} W(\mathbf{x}-\mathbf{x}_1)\mathbf{p}(\mathbf{x}_1) & \dots & W(\mathbf{x}-\mathbf{x}_n)\mathbf{p}(\mathbf{x}_n) \end{bmatrix} \in \mathbf{M}(m\times n) && (3.9)\\
\mathbf{U}(\mathbf{x}) &= \begin{bmatrix} u_1 \\ \vdots \\ u_n \end{bmatrix} \in \mathbf{M}(n\times 1). && (3.10)
\end{aligned}$$

The unknown vector **a** can now be determined as

$$\mathbf{a}(\mathbf{x}) = \mathbf{A}^{-1}(\mathbf{x})\mathbf{B}(\mathbf{x})\mathbf{U}(\mathbf{x}). \qquad (3.11)$$

By inserting this expression in 3.1, we get a new formulation of the displacement field

$$\hat{u}(\mathbf{x}) = \mathbf{p}^T(\mathbf{x})\mathbf{a}(\mathbf{x}) = \underbrace{\mathbf{p}^T(\mathbf{x})\mathbf{A}^{-1}(\mathbf{x})\mathbf{B}(\mathbf{x})}_{\phi(\mathbf{x})}\mathbf{U}(\mathbf{x}). \qquad (3.12)$$

So the displacement in a point **x** are approximated as a sum of shape functions multiplied with respectively displacement, it can be noted that $\hat{u}_i \neq u_i$. That is the consequence when the shape functions does not fulfill the Kronecker delta property. If you want to know the displacement in a point, you have to construct a support domain and compute the sum with displacement multiplied with shape function values.

3.3 Choice of support domain and weight function

There is no difference if circular or rectangular support domain are used in the EFG method. The following implementations is made with circular. The weight function plays an important role for the EFG method. A proper constructed weight function will give unique solutions when we determine the coefficient vector **a**. A weight function need to have following the properties:

- Compact support, i.e. zero outside the support domain.
- Adopt positive values for all points in the support domain
- Has its maximum value at the current point and decrease when moving outwards.

There are many kinds of function satisfying these properties, but the one used in this paper are the quartic spline function, found in for example Chen *et al.* (2006),

$$W(s_I) = \begin{cases} 1 - 6s_I^2 + 8s_I^3 - 3s_I^4, & s_I \leq 1 \\ 0, & s_I > 1. \end{cases} \tag{3.13}$$

where

$$\begin{aligned} s_I &= \frac{R_I}{\rho_I} && (3.14) \\ \rho_I &= \text{Radius of the support domain} && (3.15) \\ R_I &= \|\mathbf{r}_I\| && (3.16) \\ \mathbf{r}_I &= \mathbf{x} - \mathbf{x}_I && (3.17) \end{aligned}$$

A 2D plot of the quartic spline function is given in Figure 3.1. For details how to

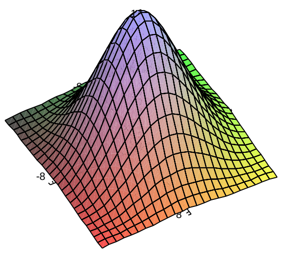

Figure 3.1: A quartic spline weight function.

construct a proper weight function see for example Liu and Liu (2003).

In the derivation of the shape functions in section 3.2 the weight function are differentiated with respect to $\mathbf{x}$. With help of the chainrule we get

$$\frac{\partial W_I}{\partial \mathbf{x}} = \frac{\partial W_I}{\partial s_I}\frac{\partial s_I}{\partial \mathbf{x}} = \begin{cases} (-12s_I + 24s_I^2 - 12s_I^3)\mathbf{r}_I/R_I\rho_I, & s_I \leq 1 \\ 0, & s_I > 1. \end{cases} \tag{3.18}$$

It can be observed that when $\mathbf{x}_I$ approach $\mathbf{x}$ in equation 3.18, both the nominator and the denominator approach zero. It is not trivial to see the limit for the problem, therefore a plot of the scalar coefficient before $\mathbf{r}_I$ in 3.18 is given in Figure 3.2. As seen in the figure the scalar term approach zero, therefore the limit also have to approach zero.

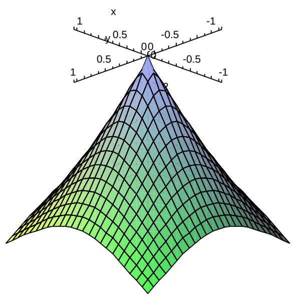

Figure 3.2: Plot of the scalar term in front of the gradient to the weight function.

3.4 Implementation

In order to calculate the matrices needed to solve the problem, a numerical integration has to be performed. For that purpose a background mesh is generated. The background mesh is completely independent of the node mesh, so it can be constructed arbitrary. In this example gauss integration is used, and each integration cell is rectangular. So instead of looping over all elements as in FEM, here it is a loop over all integration points.

To find all nodes inside the support domain, a search algorithm has to be done. There are many advanced algorithms available, but my implementation contains none of them. I just calculate the distance to all nodes. The sub routine to construct shape functions are easily implemented given the matrix formulation. The matrix inversion my be given some extra attention. A LU factorization can be done to speed it up, but this is not done in my code. Then some post processing has to be done in order to calculate the real displacement. So again you need to do a search algorithm and calculate shape functions for these points. Over all the implementation are very similar to that in FEM. In Algorithm 1 pseudocode for a general mesh free program is presented.

3.5 Numerical Example

As a first challenge to understand the EFG method, a program for linear static was implemented. A beam in bending found in Example 6.2 from Liu (2003). A figure of the beam is given in Figure 3.3.

Algorithm 1 General algorithm for EFG with Gauss integration.

```
Pre processing
-generate nodes
-generate integration points and cells
-set variables and constants
Main program
for i=1..all integration points do
  for j=1..all nodes do
    if j $\in$ support domain for i then
      -calculate shape function
    end if
  end for
  -calculate local system matrices $\boldsymbol{K}$,$\boldsymbol{G}$,$\boldsymbol{q}$ and $\boldsymbol{F}$
  -assemble to global system matrices
end for
-solve equations system
Post processing
for i=1..number of output points do
  -determine nodes inside support domain to i
  -calculate real displacement
  -compute strains and stresses for i
end for
```

The parameters given for this example are

$$\begin{array}{rclll} P &=& -1000 \text{ N} & \text{Force} & \\ E &=& 3\times 10^7 \text{ N/mm}^2 & \text{Young's modulus} & \\ \nu &=& 0.3 & \text{Poisson's ratio} & . \\ D &=& 12 \text{ m} & \text{Height of beam} & \\ L &=& 48 \text{ m} & \text{Length of beam} & \end{array}$$

For the constitutive relation we assume plane stress, so the matrix **c** is given by

$$\mathbf{c} = \frac{E}{(1-\nu^2)} \begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & (1-\nu)/2 \end{bmatrix}. \tag{3.19}$$

The advantage of using a beam example is that the analytical solution is known. So we can check the how accurate the numerical method is. The solution was made by Timonshenko and Goddier (1970),

$$u_x = -\frac{P}{6EI}\left((6L-3x)x + (2+\nu)\left(y^2 - \frac{D^2}{4}\right)\right) \tag{3.20}$$

$$u_y = \frac{P}{6EI}\left(3\nu y^2(L-x) + (4+5\nu)\frac{D^2x}{4} + (3L-x)x^2\right). \tag{3.21}$$

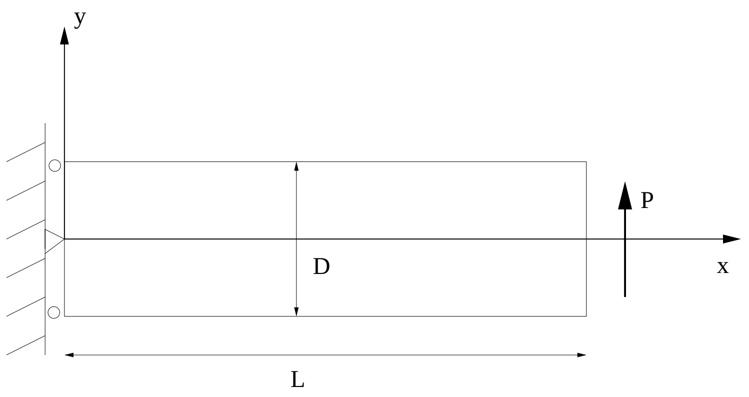

Figure 3.3: Cantilever beam load by a vertical force at in it free end

also the exact stresses where calculated by Timoschenko and Goodier

$$\sigma_{xx} = -\frac{P(L-x)y}{I} \tag{3.22}$$

$$\sigma_{yy} = 0 \tag{3.23}$$

$$\sigma_{xy} = -\frac{P}{2I}\left(\frac{D^2}{4} - y^2\right). \tag{3.24}$$

The traction vector at the free end (x=48) are parabolic and given by

$$t = \frac{P}{2I}\left(\frac{D^2}{4} - y^2\right). \tag{3.25}$$

The moment of inertia for a rectangular surfaces with unit thickness is

$$I = \frac{D^3}{12}.$$

The displacement are prescribed at the boundary (x=0). The magnitude are easily calculated by just put (x=0) in 3.20 and 3.21.

The domain was divided in a number of nodes, N_x nodes in x direction and N_y nodes in the y direction. So the total number of nodes where $N = N_x \times N_y$. An example of the mesh with regular node distribution and irregular node distribution can be seen in Figure 3.4.

To perform gauss integration the domain was divided in nbrofcellsx cells in x direction and nbrofcellsy in y direction. So, as in the mesh, the total number of cells are nbrofcells = nbrofcellsx $\times$ nbrofcellsy. Each cell contains pointspercell points where the functionvalues are evaluated, the total number of integration points where then nbrofcells $\times$ pointspercell.

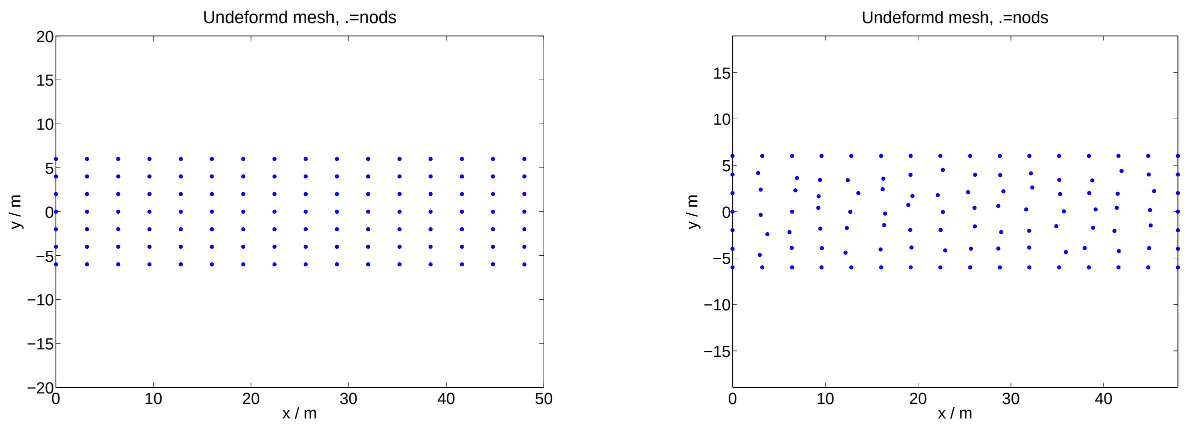

Figure 3.4: Example of node distribution with $N_x = 16$ and $N_y = 7$: a) regular b) irregular

3.6 Results for Beam in bending

The code was implemented in Matlab, and tested for several number of nodes and integration points. For each test, given the number of nodes, the number of integration points and the support domain where

$$\begin{aligned}\text{nbrofcellsx} &= N_x \times 2\\ \text{nbrofcellsy} &= N_y \times 2\\ \text{pointspercell} &= 8 \times 8\\ \text{support domain} &= 2\sqrt{\frac{D^2}{(N_y-1)^2} + \frac{L^2}{(N_x-1)^2}}.\end{aligned}$$

In each test the relative error of the displacement in y direction for $x = 48$ and $y = 0$ was calculated as

$$\frac{u_{analytical} - u_{EFG}}{u_{EFG}}.$$

The results for both regular and irregular nodes are given in Table 3.1.

nodes($N_x \times N_y$)	relative error regular(%)	relative error irregular(%)
7 × 5	2.4992	2.5901
11 × 5	1.1950	1.1838
16 × 7	0.8717	0.8708
20 × 9	0.7606	0.7835
30 × 14	0.6305	0.6376
60 × 29	0.5369	0.5566

Table 3.1: Relative error for the EFG method, applied to a cantilever beam, for both regular and irregular mesh.

The relative error converge to zero when the number of nodes approach infinity. The analytical stresses where also known for the problem, so for the last case ($N_x = 60$ and $N_y = 29$) the stresses where calculated in a number of points for the section $x = 24$. Figure 3.5 contains the deflection of the beam for $x = 0$ and the normal stress in x-direction for the section and Figure 3.6 contains the normal stress in y-direction and shear stress for the section.

For regular node distribution we have excellent agreement, for both displacement and stresses. For the case with irregular nodes, the displacement also seems very accurate. But the normal stress in y contains some errors, but not very large compared with the magnitude of the normal stress in x. The problem with irregular nodes are that for the current node the support domain to the left contains much more nodes than to the right. Therefore it gets a little bit unbalanced. The main reason when having irregular mesh, is that you want more nodes at some location, but on that location the distance between the nodes is constant. Then you don not have this problem. Another solution would have been to use some kind of adaptive support domain.

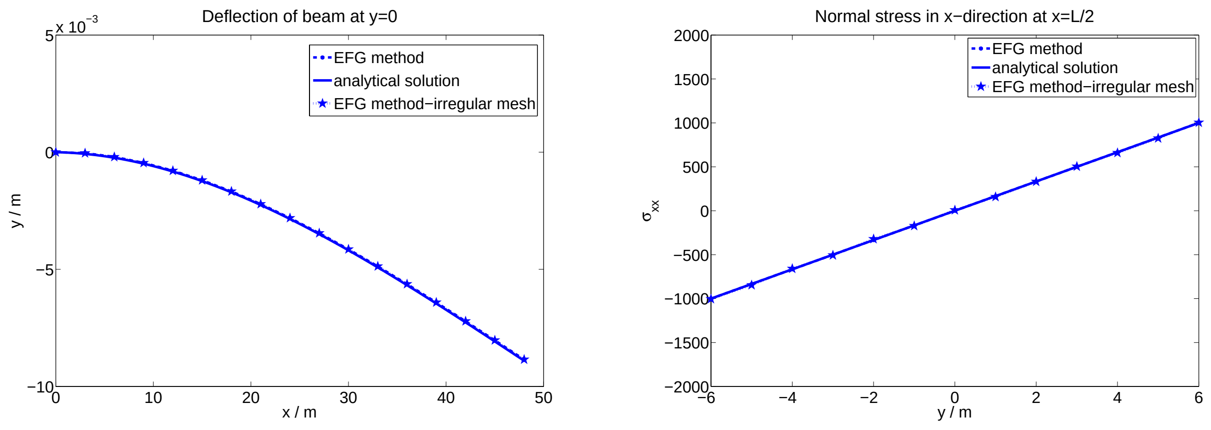

Figure 3.5: Comparison of the test result for the cantilever beam, both with regular mesh and irregular, and with $N_x = 60$ and $N_y = 29$: a) deflection in y at y=0 b) normal stress in x at x=L/2

3.7 Summary

An Element free Galerkin method was implemented in Matlab for linear statics. The implementation was straightforward, the only thing that needed some extra thought was the shape functions. But they where easily implemented given the formulation from section 3.2. The assembling part and the whole structure of the program looks like a regular finite element program.

The method seems accurate enough and converge to the analytical solution when the number of nodes approach infinity. A little remark has to be mentioned about

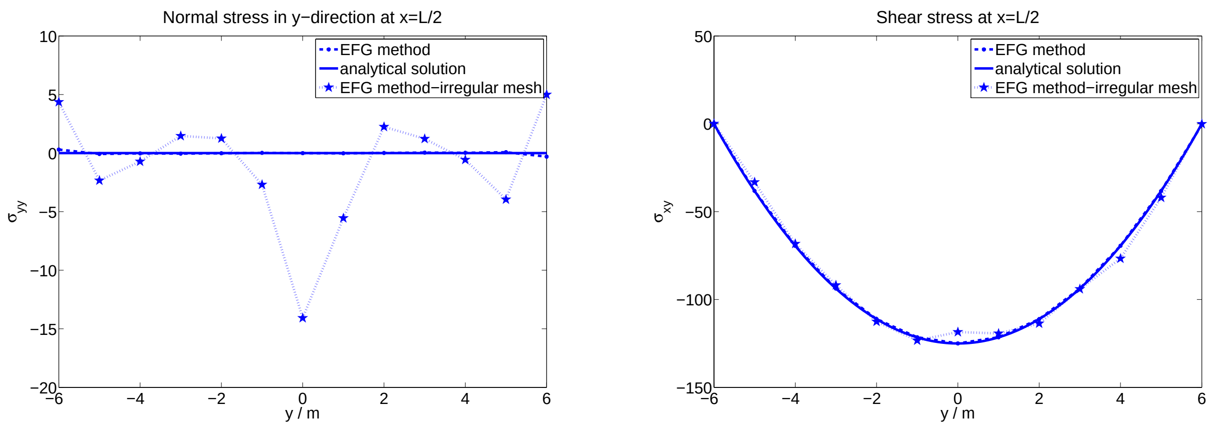

Figure 3.6: Comparison of the test result for the cantilever beam, both with regular mesh and irregular, and with $N_x = 60$ and $N_y = 29$: a) normal stress in y at x=L/2 b) shear stress at x=L/2

the numerical integration. To have accurate result, the numerical integration has to be very fine, i.e. many integration points. Note that it converge even with a small amount of integration points, but it is not accurate enough. The only requirement to converge is that the number of nodes in the support domain is greater than the number of components in $\mathbf{p}$. This to guarantee that the matrix $\mathbf{A}$ is invertible.

The many integration points combined with the enlarged system because of the Lagrange multiplier, results in a very time consuming program. Even for a so simple problem with linear statics. But no optimization where made on the code to improve the performance.

Chapter 4

Reproducing Kernel Particle Method

4.1 Introduction

Another mesh free method you find in the literature is the Reproducing Kernel Particle Method(RKPM) developed by Liu *et al.* (1995). This method has been successfully applied to many different kind of problems including large deformations. There is no difference in how to construct the weak form compared to EFG, the main difference is the shape functions.

In EFG we approximated the displacement $\mathbf{u}$ in a point $\mathbf{x}$ by a sum of all nodes that lies in the support domain of $\mathbf{x}$ multiplied with the value of the shape function in that point. In RKPM we end up with a similar expression but the shape functions differs from EFG. There is a significant difference how to develop the shape functions in these methods, EFG starts the approximation with a serial representation and RKPM starts with a integral representation. RKPM is motivated by the theory of wavelets where the function is represented by a combination of dilation and translation of a single wavelet. This often leads to an integral representation of the function itself.

4.2 Deriving shape functions

The RKPM approximation starts from an integral representation of the unknown function $\mathbf{u}$

$$\mathbf{u}(\mathbf{x}) = \int_{\Omega_I} \mathbf{u}(\tilde{\mathbf{x}})\mathrm{K}(\mathbf{x}, \tilde{\mathbf{x}})\ d\tilde{V} \tag{4.1}$$

where $\mathbf{K}$ is the kernel function. The integral is only defined over the supportdomain Ω_I due to the compact suppport of the weight function. It is trivial to see that if $\mathrm{K}(\mathbf{x}) \rightarrow \delta(\mathbf{x})$ this integral transformation should generate the exact displacement. So the problem is to choose Kernel functions that mimics a Kronecker delta.

In the Smoothed Particle Hydrodynamics method, you simply choose the kernel as the weight function. But Liu and co-workers discovered that this method had problems with the consistency at the boundaries. Consistency for mesh free methods can compared to completeness for FEM. It is the ability for the approximation to exactly reproduce a polynomial of certain order. For the case of linear consistency it becomes

$$\mathbf{c} = \int_{\Omega_I} \mathbf{c}\mathrm{K}(\mathbf{x},\tilde{\mathbf{x}})\ d\tilde{V} = \mathbf{c}\int_{\Omega_I} \mathrm{K}(\mathbf{x},\tilde{\mathbf{x}})\ d\tilde{V} \tag{4.2}$$

$$\mathbf{x} = \int_{\Omega_I} \tilde{\mathbf{x}}\mathrm{K}(\mathbf{x},\tilde{\mathbf{x}})\ d\tilde{V}. \tag{4.3}$$

In order to obtain desired consistency over the entire domain, he introduced a kernel function that where the weight function multiplied with a correction function. And this correction function is constructed just to fulfill the consistency condition.

4.2.1 Construction of correction function

If we choose to have linear consistency then the correction function will take the form

$$K(\mathbf{x},\tilde{\mathbf{x}}) = C(\mathbf{x},\tilde{\mathbf{x}})W(\mathbf{x}-\tilde{\mathbf{x}}) = (C_0(\mathbf{x}) + \mathbf{C}_1(\mathbf{x})(\mathbf{x}-\tilde{\mathbf{x}}))W(\mathbf{x}-\tilde{\mathbf{x}}). \tag{4.4}$$

It can be seen that the correction function is linear with respect to $\mathbf{x} - \tilde{\mathbf{x}}$. The consistency conditions implies that the approximation is able to reproduce a constant and a linear term exactly. For the first consistency, equation 4.2, to be fulfilled the integral over the kernel function have to be equal to one. So the first consistency condition can be reformulated as

$$\int_{\Omega_I} \mathrm{K}(\mathbf{x},\tilde{\mathbf{x}})\ d\tilde{V} = 1 \tag{4.5}$$

which is also called the normalization property.

The second consistency condition, equation 4.3, also have to be rewritten in order to fit with the theory

$$\mathbf{x} = \int_{\Omega_I} \tilde{\mathbf{x}}\mathrm{K}(\mathbf{x},\tilde{\mathbf{x}})\ d\tilde{V} = \int_{\Omega_I} \mathbf{x}\mathrm{K}(\mathbf{x},\tilde{\mathbf{x}})\ d\tilde{V} - \int_{\Omega_I} (\mathbf{x}-\tilde{\mathbf{x}})\mathrm{K}(\mathbf{x},\tilde{\mathbf{x}})\ d\tilde{V} = \tag{4.6}$$

$$= \mathbf{x}\underbrace{\int_{\Omega_I} \mathrm{K}(\mathbf{x},\tilde{\mathbf{x}})\ d\tilde{V}}_{=1} - \int_{\Omega_I} (\mathbf{x}-\tilde{\mathbf{x}})\mathrm{K}(\mathbf{x},\tilde{\mathbf{x}})\ d\tilde{V} = \mathbf{x} - \int_{\Omega_I} (\mathbf{x}-\tilde{\mathbf{x}})\mathrm{K}(\mathbf{x},\tilde{\mathbf{x}})\ d\tilde{V}. \tag{4.7}$$

Here we used the normalization property of the kernel function. For this equation to be fulfilled, the second term has to be zero. So the second consistency condition can again be reformulated as

$$\int_{\Omega_I} (\mathbf{x}-\tilde{\mathbf{x}})\mathrm{K}(\mathbf{x},\tilde{\mathbf{x}})\ d\tilde{V} = \mathbf{0}. \tag{4.8}$$

Before we use this expressions to determine the **c** coefficients, we start with define some quantities named moments

$$
\begin{aligned}
m_0(\mathbf{x}) &= \int_{\Omega_I} W(\mathbf{x}-\tilde{\mathbf{x}})\, d\tilde{V} && (4.9)\\
\mathbf{m}_1(\mathbf{x}) &= \int_{\Omega_I} (\mathbf{x}-\tilde{\mathbf{x}})W(\mathbf{x}-\tilde{\mathbf{x}})\, d\tilde{V} && (4.10)\\
\mathbf{m}_2(\mathbf{x}) &= \int_{\Omega_I} (\mathbf{x}-\tilde{\mathbf{x}})(\mathbf{x}-\tilde{\mathbf{x}})^T W(\mathbf{x}-\tilde{\mathbf{x}})\, d\tilde{V}. && (4.11)
\end{aligned}
$$

By inserting our modified kernel function 4.4 in the first consistency condition 4.5 and use our defined moments we end up with

$$
1 = \int_{\Omega_I} (C_0(\mathbf{x}) + \mathbf{C}_1(\mathbf{x})(\mathbf{x}-\tilde{\mathbf{x}}))W(\mathbf{x}-\tilde{\mathbf{x}})\, d\tilde{V} = C_0(\mathbf{x})m_0(\mathbf{x}) + \mathbf{C}_1(\mathbf{x})\mathbf{m}_1(\mathbf{x}). \quad (4.12)
$$

And in the same manner equation 4.8 gives us

$$
\begin{aligned}
\mathbf{0} &= \int_{\Omega_I} ((\mathbf{x}-\tilde{\mathbf{x}})(C_0(\mathbf{x}) + \mathbf{C}_1(\mathbf{x})(\mathbf{x}-\tilde{\mathbf{x}}))W(\mathbf{x}-\tilde{\mathbf{x}})\, d\tilde{V} = \\
&= C_0(\mathbf{x})\mathbf{m}_2(\mathbf{x}) + \mathbf{C}_1(\mathbf{x})\mathbf{m}_2(\mathbf{x}). && (4.13)
\end{aligned}
$$

If we define ND as number of dimension, and write the two equations as an equation system, we get

$$
\begin{bmatrix} m_0(1\times 1) & \mathbf{m}_1^T(1\times ND) \\ \mathbf{m}_1(ND\times 1) & \mathbf{m}_2(ND\times ND) \end{bmatrix}
\begin{bmatrix} C_0(1\times 1) \\ \mathbf{C}_1(ND\times 1) \end{bmatrix}
=
\begin{bmatrix} 1(1\times 1) \\ \mathbf{0}(ND\times 1) \end{bmatrix}. \quad (4.14)
$$

This can in a compact form be written as

$$
\mathbf{MC} = \mathbf{H}(\mathbf{0}). \quad (4.15)
$$

Here $\mathbf{M}$ is the moment matrix, $\mathbf{C}$ the wanted coefficient vector and $\mathbf{H}(\mathbf{0}) = [1\ 0\ 0]^T$. Given the moment matrix it is now possible to determine the unknown coefficients just by inverting it. As seen in the next section, when differentiating the shape functions, it is also necessary to know the differentiated coefficients. They can be determined by differentiate equation 4.15, for two dimensions it becomes

$$
\begin{aligned}
\frac{\partial \mathbf{M}}{\partial x}\mathbf{C} + \mathbf{M}\frac{\partial \mathbf{C}}{\partial x} &= \mathbf{0} && (4.16)\\
\frac{\partial \mathbf{M}}{\partial y}\mathbf{C} + \mathbf{M}\frac{\partial \mathbf{C}}{\partial y} &= \mathbf{0} && (4.17)
\end{aligned}
$$

We are now ready to formulate the final equation system when determine the coefficients

$$
\begin{bmatrix} \mathbf{M} & \mathbf{0} & \mathbf{0} \\ \frac{\partial \mathbf{M}}{\partial x} & \mathbf{M} & \mathbf{0} \\ \frac{\partial \mathbf{M}}{\partial y} & \mathbf{0} & \mathbf{M} \end{bmatrix}
\begin{bmatrix} \mathbf{C} \\ \frac{\partial \mathbf{C}}{\partial x} \\ \frac{\partial \mathbf{C}}{\partial y} \end{bmatrix}
=
\begin{bmatrix} \mathbf{H}(\mathbf{0}) \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix}. \quad (4.18)
$$

When you have constructed the enlarged moment matrix, it is just to invert it and then you have the coefficients and derivatives.

4.2.2 Discretization of integral

Now the kernel function in equation 4.1 is known, but in order to implement the approximation on a computer, we have to discretize the integral. If the support domain Ω_I contains n nodes the integral is approximated as

$$\mathbf{u}(\mathbf{x}) \approx \sum_{j=1}^{n} u_j C(\mathbf{x}, \mathbf{x}_j) W(\mathbf{x} - \mathbf{x}_j) \Delta V_j = \sum_{j=1}^{NP} \phi_j u_j \tag{4.19}$$

where ΔV_j is the volume associated with node j. The sum of all ΔV_j have to be the complete volume V.

It is easy to obtain the first derivate of the shape function

$$\phi_{j,x} = (C_{,x}(\mathbf{x}, \mathbf{x}_j) W(\mathbf{x} - \mathbf{x}_j) + C(\mathbf{x}, \mathbf{x}_j) W_{,x}(\mathbf{x} - \mathbf{x}_j)) \Delta V_j. \tag{4.20}$$

4.2.3 Summary

Given a point $\mathbf{x}$ this is how you construct shape functions for all nodes in the support domain.

- determine the support domain
- integrate over the support domain and calculate the moments m_0, $\mathbf{m}_1$, $\mathbf{m}_2$ and it is derivatives. Determine the coefficients vector $\mathbf{c}$.
- find all nodes in the support domain
- for all nodes calculate the value of the weight function and after that the shape function

It can be noted that the coefficients $\mathbf{c}$ is completely independent of the coordinate for the node point. It was also shown by Liu *et al.* (1995) that if $\mathbf{x}$ is far away from the boundary, the coefficients becomes $C_0 = 1$ and $\mathbf{C}_1 = [0\ 0]^T$. So when the support domain is completely inside the problem domain the shape function value just becomes the weight function value multiplied with the corresponding volume, exactly as in the Smooth Particle Hydrodynamics method.

If you want a higher order approximation, you just add extra terms in the correction function $C(\mathbf{x}, \tilde{\mathbf{x}})$ in equation 4.4, which will give extra coefficients. But the consistency conditions will also get more constraints, which will lead to a enlarged moment matrix. So it will only lead to a larger equation system to solve when determine the coefficients.

4.3 Choice of support domain and weight function

In the EFG method we had a circular support domain, and a weight function with only one variable, the radius. But now, when we need to integrate over the support domain to get our moment matrices, it is handy to have a rectangular support domain. Because these integrals have to be integrated numerically and the division is much easier with a rectangular support domain.

The weight function for a point $\mathbf{x}$ in one variable is generalized to two as

$$W(s_x, s_y) = \begin{cases} (1 - 6s_x^2 + 8s_x^3 - 3s_x^4)(1 - 6s_y^2 + 8s_y^3 - 3s_y^4)\frac{25}{16}, & s_x \leq 1 \text{ and } s_y \leq 1 \\ 0, & s_x > 1 \text{ or } s_y > 1 \end{cases} \tag{4.21}$$

where

$$s_x = \frac{|x - \tilde{x}|}{R_x} \tag{4.22}$$

$$s_y = \frac{|y - \tilde{y}|}{R_y} \tag{4.23}$$

An integral of the weight function over the support domain, independent of the size, always becomes $\frac{16}{25}$. That is why we have the constant term in the weight function, just to normalize it. For the EFG method this property is irrelevant, but for the RKPM it is important because of the first consistency condition. As mentioned before, in the body the kernel function just becomes the weight function. So in order to equation 4.5 to be fulfilled the weight function has to be normalized.

The first derivate is constructed exactly as equation 3.18 and becomes

$$\frac{\partial W}{\partial x} = \begin{cases} (-12s_x + 24s_x^2 - 12s_x^3)(1 - 6s_y^2 + 8s_y^3 - 3s_y^4)(x - \tilde{x})25/(s_x R_x 16), & s_x \leq 1 \\ 0, & s_x > 1. \end{cases} \tag{4.24}$$

$$\frac{\partial W}{\partial y} = \begin{cases} (-12s_y + 24s_y^2 - 12s_y^3)(1 - 6s_x^2 + 8s_x^3 - 3s_x^4)(y - \tilde{y})25/(s_y R_y 16), & s_y \leq 1 \\ 0, & s_y > 1. \end{cases} \tag{4.25}$$

As for the weight function in one variable it hold that if s_x or s_y approach zero, the differentiated weight function also approach zero.

4.4 Implementation

The program looks exactly like the EFG program, except the routine that returns the shape functions. Implementation of this routine is slightly more complex than for EFG. You have to calculate a volume corresponding to each node when discretizing the integral and you also have to perform a numerical integration over the support domain to get the moment matrices.

This in combination with a bunch of derivates to be calculated made the programming kind of tricky. But there are several checks that can be done to investigate the reliability of the code, for example that the correction term becomes one when the support domain is completely inside the problem domain and the the integral over the kernel always becomes one, even when we are close to the boundary.

4.5 Results for Beam in bending

An RKPM program where implemented for the exact same test as the EFG program. In this program we had a rectangular support domain, like for the EFG method, the size depends on how many nodes there are

$$\begin{aligned} \mathsf{R}_x &= 4 \times \frac{L}{N_x - 1} \\ \mathsf{R}_y &= 4 \times \frac{D}{N_y - 1}. \end{aligned}$$

This will approximately give the same number of nodes in the support domain and except these two parameters everything are the same, see section 3.5 for details.

As for the EFG program the implementation where tested for different number of nodes, both with regular and irregular node distribution. For each test the relative error where calculated in the point ($x = 48$,$y = 0$) as

$$\frac{u_{analytical} - u_{RKPM}}{u_{RKPM}}. \tag{4.26}$$

The results are collected in Table 4.1. The relative error converge to zero when the number of nodes approach infinity. This holds for both regular and irregular node distribution.

nodes($N_x \times N_y$)	relative error regular(%)	relative error irregular(%)
7 × 5	1.6227	1.6140
11 × 5	0.9694	0.9648
16 × 7	0.7401	0.7507
20 × 9	0.6623	0.6790
30 × 14	0.5794	0.5816
60 × 29	0.5193	0.5329

Table 4.1: Relative error for the RKPM method, applied to a cantilever beam, for both regular and irregular mesh.

To be able to compare the both methods, the displacement and stresses are plotted exactly as for the EFG example. The displacement for $y = 0$ and the normal stress in x-direction for $x = L/2$ are plotted in Figure 4.1. And the normal stress in y-direction and the shear stress for $x = L/2$ are plotted in Figure 4.2. The results shows excellent agreement of the displacement for both regular and irregular node distribution. The same holds for the normal stress in x-direction, but exactly as for the EFG example the normal stress in y-direction contains some errors, especially with irregular nods. But the error is not bigger than for the EFG method and not large compared to the stresses in the x-direction. The shear stress also seems accurate enough except for some parts where the solution with irregular nods are a little bit big. This depends on a large disorientation of the nods a this point. A magnified support domain should have solved the problem.

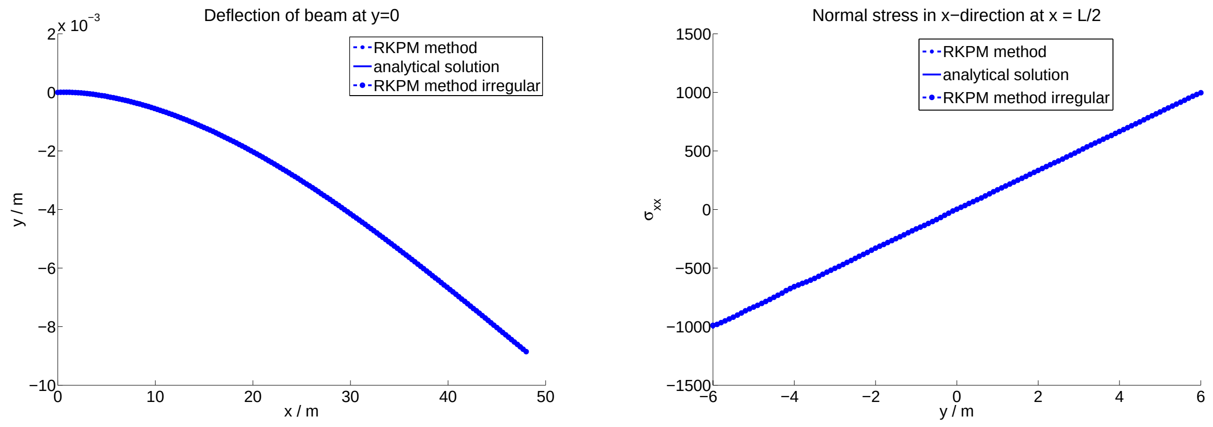

Figure 4.1: Comparison of the test result for the cantilever beam, both with regular and irregular mesh, and with $N_x = 60$ and $N_y = 29$: a) deflection in y at y=0 b) normal stress in x at x=L/2

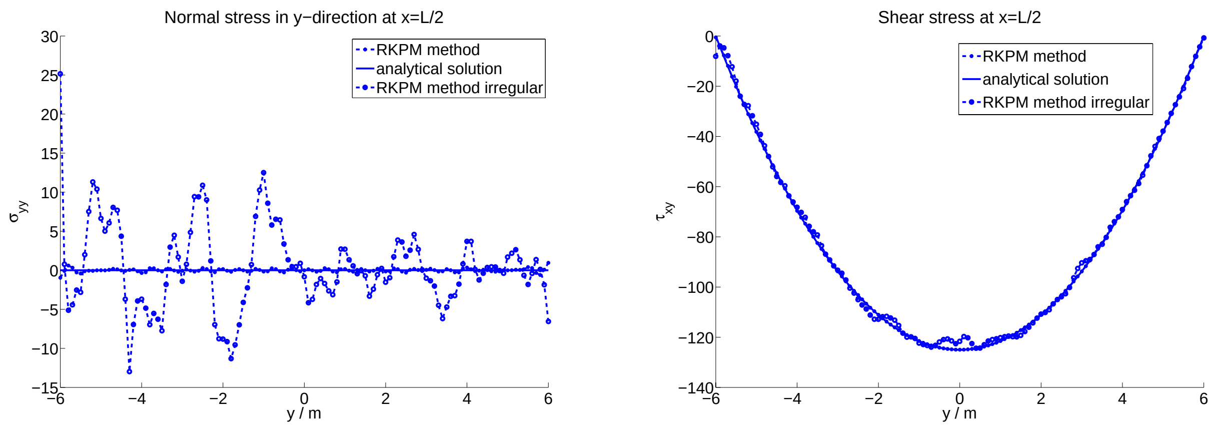

Figure 4.2: Comparison of the test result for the cantilever beam, both with regular and irregular mesh, and with $N_x = 60$ and $N_y = 29$: a) normal stress in y at x=L/2 b) shear stress at x=L/2

4.6 Summary

The RKPM method seems to give accurate results and converge when the number of nodes approach zero. Also this program is time consuming for such a small problem as this. If you compare the two methods for this simple problem there is almost no difference. The convergence is the same, the stress field very similar and both method seems to have problems with the same thing, i.e. the normal stress in y-direction. The time spend in creating shape functions are also the same. For both methods the inversion of the moment matrix respective **A** matrix is the most time consuming part. And if the number of basis coefficient are the same, also the size of the matrices are the same. As a matter of fact, if the volume per node is chosen to one, $\Delta V_i = 1$, then RKPM give the exact same shape functions as MLS method. This was shown by Liu *et al.* (1997).

Chapter 5

RKPM for plasticity problem

5.1 Theory

To further test our mesh free program, a plasticity model where implemented. For the linear case, there was almost no difference of the performance between EFG and RKPM. Therefore only the RKPM where further developed. Also the fact that many existing articles handling large deformation problem use RKPM played a significant role.

5.1.1 Mesh free formulation

In equation 2.10 a linear elastic constitutive relation was needed in order to manipulate the equation. Without further proof this can be done for a plastic constitutive relation so again we end up with equation 2.15. For now we only consider small deformations, therefore the strain can, as before, be written as $\varepsilon = \mathbf{BU}$. Inserted in equation 2.15 gives with definitions given in section 2.3

$$\delta\mathbf{U}^T(\int_\Omega \mathbf{B}^T\boldsymbol{\sigma}\ d\Omega - \mathbf{f} + \mathbf{G}\boldsymbol{\Lambda}) + \delta\boldsymbol{\Lambda}^T(\mathbf{G}^T\mathbf{U} - \mathbf{q}) = \mathbf{0}. \tag{5.1}$$

The variations can not always be equal to zero, and they are independent, therefore the terms they are multplied with have to be equal to zero. This results in a equation system

$$\boldsymbol{\Psi}(\mathbf{U}, \boldsymbol{\Lambda}) = \left\{ \begin{array}{rcl} \int_\Omega \mathbf{B}^T\boldsymbol{\sigma}\ d\Omega - \mathbf{f} + \mathbf{G}\boldsymbol{\Lambda} & = & \mathbf{0} \\ \mathbf{G}^T\mathbf{U} - \mathbf{q} & = & \mathbf{0}. \end{array} \right. \tag{5.2}$$

This is the equilibrium equations for the plasticity problem that we want to fulfill in every step. Because the non-linear constitutive relation, we have to iterate in order to find this equilibrium. This is done by a Newton-Raphson algorithm. Without going through any details this is done by linearize the equation around a given point, put it equal to zero and then solve the linear equation system to get our new variables. Then check if the new variables fulfill the equilibrium equations.

Starting from a point $i-1$ where $\mathbf{U}_{i-1}$ and $\boldsymbol{\Lambda}_{i-1}$ is known and i stands for number of iterations. The linearization is done by a Taylor expansion around $i-1$ where higher order terms are neglected. The following equation is obtained

$$\boldsymbol{\Psi}(\mathbf{U}_{i-1}, \boldsymbol{\Lambda}_{i-1}) + \boldsymbol{\Psi}_{\mathbf{U}}|_{i-1}(\mathbf{U}_i - \mathbf{U}_{i-1}) + \boldsymbol{\Psi}_{\boldsymbol{\Lambda}}|_{i-1}(\boldsymbol{\Lambda}_i - \boldsymbol{\Lambda}_{i-1}) = 0. \tag{5.3}$$

Before proceeding with differentiation of the equilibrium equations, a modification of the constitutive equation have to be made. For a plasticity problem the stress and strains are related as $\dot{\boldsymbol{\sigma}} = \mathbf{D}\dot{\boldsymbol{\varepsilon}}$. This can be modified as follows

$$\dot{\boldsymbol{\sigma}} = \mathbf{D}\dot{\boldsymbol{\varepsilon}} \Leftrightarrow d\boldsymbol{\sigma} = \mathbf{D}d\boldsymbol{\varepsilon} \Leftrightarrow d\boldsymbol{\sigma} = \mathbf{D}\mathbf{B}d\mathbf{U} \Leftrightarrow \frac{d\boldsymbol{\sigma}}{d\mathbf{U}} = \mathbf{D}\mathbf{B}. \tag{5.4}$$

If we use this result when differentiate the upper row in the equilibrium equation with respect to $\mathbf{U}$, we get

$$\int_{\Omega} \mathbf{B}^T \frac{d\boldsymbol{\sigma}}{d\mathbf{U}}\, d\Omega = \int_{\Omega} \mathbf{B}^T\mathbf{D}\mathbf{B}\, d\Omega = \mathbf{K}. \tag{5.5}$$

The derivatives in equation 5.3 now becomes

$$\boldsymbol{\Psi}_{\mathbf{U}} = \frac{d\boldsymbol{\Psi}}{d\mathbf{U}} = \begin{cases} \mathbf{K} \\ \mathbf{G}^T \end{cases} \tag{5.6}$$

$$\boldsymbol{\Psi}_{\boldsymbol{\Lambda}} = \frac{d\boldsymbol{\Psi}}{d\boldsymbol{\Lambda}} = \begin{cases} \mathbf{G} \\ \mathbf{0}. \end{cases} \tag{5.7}$$

For simplicity a new variable is introduced

$$\mathbf{a} = \begin{bmatrix} \mathbf{U} \\ \boldsymbol{\Lambda} \end{bmatrix}. \tag{5.8}$$

Then equation 5.3 becomes

$$\boldsymbol{\Psi}(\mathbf{a}_i) + \underbrace{\begin{bmatrix} \mathbf{K} & \mathbf{G} \\ \mathbf{G}^T & \mathbf{0} \end{bmatrix}}_{\text{Iteration matrix}} (\mathbf{a}_i - \mathbf{a}_{i-1}) = \mathbf{0}. \tag{5.9}$$

This is the mesh free formulation for a plasticity problem, from this we can calculate new field variables by inverting the iteration matrix. It can be noted that iteration matrix takes exactly the same form as the enlarged stiffness martix from the linear elastic case, except $\mathbf{D}$ is different.

5.1.2 Model

The implemented plasticity model is a linear kinematic hardening von Mises model. A summarize are given in Table 5.1 and explanation of the variables given in Table 5.2. For more details see Ristinmaa and Ottosen (2005).

$$\boxed{\text{Yield conditions}}$$

$$f = \sigma_{eff} - \sigma_{y0}$$
$$\sigma_{eff} = \sqrt{(\tfrac{3}{2}\beta_{ij}\beta_{ij})}$$
$$\beta_{ij} = s_{ij} - \alpha_{ij}^{d}$$
$$s_{ij} = \sigma_{ij} - \tfrac{1}{3}\delta_{ij}\sigma_{kk}$$

$$\boxed{\text{Evolution laws}}$$

$$\dot{\varepsilon}_{ij}^{p} = \dot{\lambda}\frac{\partial f}{\partial \sigma_{ij}}$$
$$\dot{\alpha}_{ij}^{d} = c\dot{\varepsilon}_{ij}^{p}$$

Table 5.1: Summarize of the plasticity model.

variable	explanation
f	yield condition
$\mathbf{s}$	deviatoric stress tensor
σ_{y0}	yield stress
$\boldsymbol{\varepsilon}^{p}$	plastic strain
λ	plastic multiplier
c	material parameter
$\boldsymbol{\alpha}^{d}$	deviatoric part of the backstress tensor

Table 5.2: Explanation of the variables in the plasticity model.

5.2 Implementation

Given the RKPM program for linear statics and an implemented material function there where no problem to combine the two programs to one. Because we only look at small strains, the reference configuration always where the initial configuration. therefore the shape functions for all integration points was constant during the simulation. So in the beginning of the program the shape functions where stored for each integration point, and after that they where only fetched. Constructing the shape functions is the most time consuming part of the simulation, so with this technique you save a lot of time. This in combination that the F, G and q matrices where calculated outside the iteration loop, the program where relative fast. Each iteration took only a few seconds.

5.3 Results for beam in bending

The new plasticity model was tested for the now well known beam example. Parameter for the example is given in Table 5.3. The load loop consist of changing the parameter

$$K = 2.500 \cdot 10^7 \text{ MPa}$$
$$G = 1.1538 \cdot 10^7 \text{ MPa}$$
$$\sigma_{y0} = 600 \text{ MPa}$$
$$c = 25 \cdot 10^3 \text{ MPa}$$

Table 5.3: Material parameters for beam example.

P, see section 3.5 for details. The beam starts from $P = 0$ and where loaded up to $P = -500$ N, and after that unloaded to $P = 0$ again. The plastic deformations starts with approximately $P = -340$ N. Three pictures of the deflection are shown for the initial configuration, the maximum pressure and when the unloading is finished (i.e. $P = 0$) in Figure 5.1 to Figure 5.3. After the loading cycle is complete there is still some plastic deformation left, see Figure 5.3.

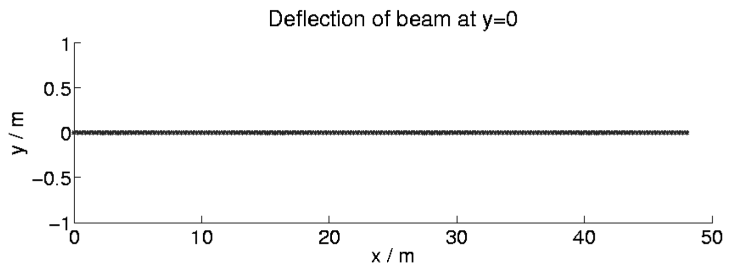

Figure 5.1: Deflection at y=0 for the beam in the initial state, $P = 0$.

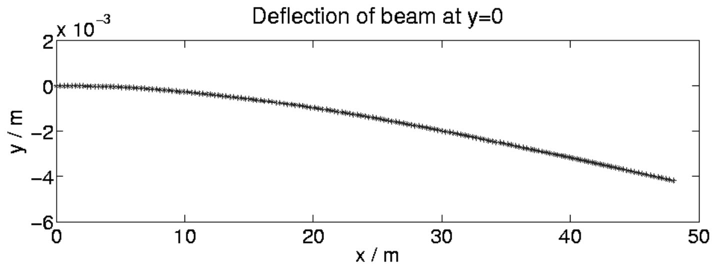

Figure 5.2: Deflection at y=0 for the beam at $P = -500$.

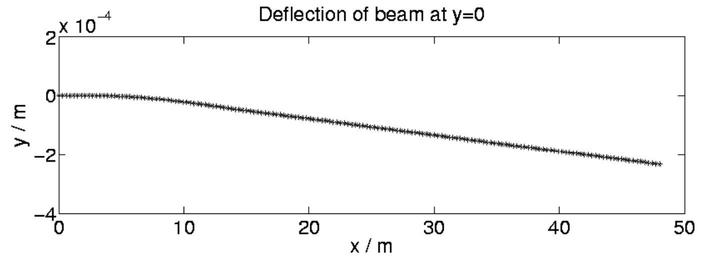

Figure 5.3: Deflection at y=0 for the beam in the initial state, after the loading cycle is complete. Some displacement still occur due to plastic deformations.

5.4 Results for Cooks membrane

To further test the implemented code, the well known Cook's membrane was also investigated. A sketch of the membrane are given in Figure 5.4. The same linear kinematic plasticity model as for the beam is used, and the material parameters are given in Table 5.4. The membrane was loaded with a constant traction vector at the right side. It was simply supported at the left side, i.e. displacement zero in both directions.

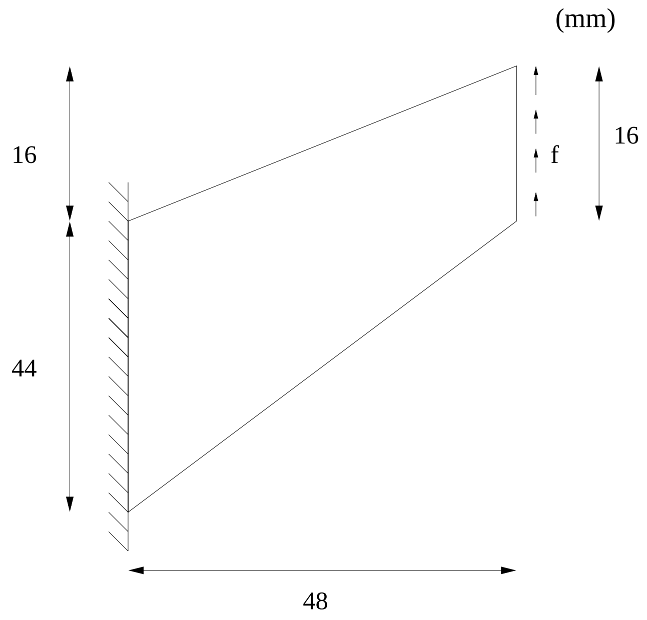

Figure 5.4: Sketch of Cooks membrane.

The membrane was tested with a force controlled cycle loading, given by $\{0 \rightarrow f_{max} \rightarrow f_{min} \rightarrow f_{max} \rightarrow f_{min} \rightarrow f_{max} \rightarrow 0\}$, where $f_{max} = 2500N$ and $f_{min} = -2500N$. Values of the load in the cycle is the total force acting on the body, not the traction

$K = 164 \cdot 10^3$ MPa
$G = 80 \cdot 10^3$ MPa
$\sigma_{y0} = 400$ MPa
$c = 18 \cdot 10^3$ MPa

Table 5.4: Material parameters for Cooks membrane.

vector. A plot of the response for the lower right node is given in Figure 5.5. After the first cycle nothing will change, this is due to the kinematic hardening of the plasticity. The plasticity occur approximately at 1.5 kN.

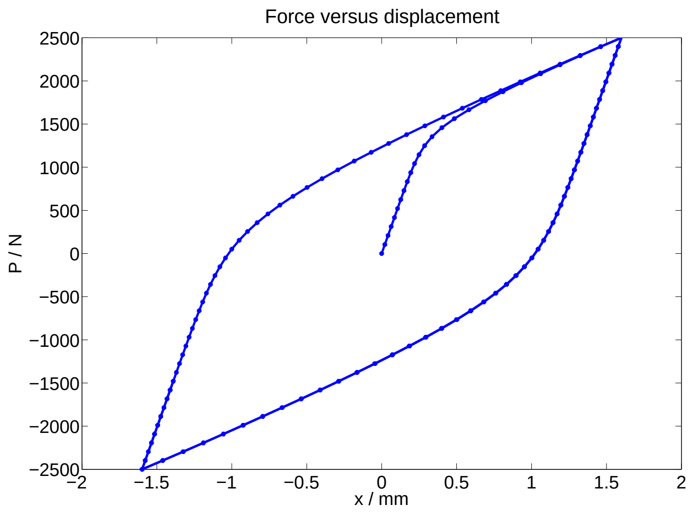

Figure 5.5: Results for Cook's membrane tested with a cycle load. Displacement versus total force for the node in the lower right corner.

To further investigate the method, a convergence test was performed on the membrane. The membrane was loaded from zero and up to f_{max} in 25 equal steps, i.e. $\Delta f = 100N$. For an increasing number of nodes, the displacement for the maximum load was calculated. The results can be seen in Table 5.5.

The method converges, as the number of nodes increase and have irregular nodes does not change the performance. It is surprisingly accurate even for very few nodes, to show that a plot of the displacement versus force for both 10×8 and 50×48 nodes are shown in Figure 5.6. Also a plot of the node distribution for the irregular case is included, both the initial configuration and the maximal deformed configuration can be seen in Figure 5.7.

The Newton iteration converges quadratically and there is no difference if there is regular or irregular nodes. Number of nodes does not either influence the conver-

	Regular nodes	Irregular nodes
	u / 2500 N	u / 2500 N
10×8	1.601	1.604
20×18	1.640	1.635
30×28	1.650	1.645
40×38	1.655	1.649
50×48	1.658	1.655

Table 5.5: Convergence of displacement for both regular and irregular nodes.

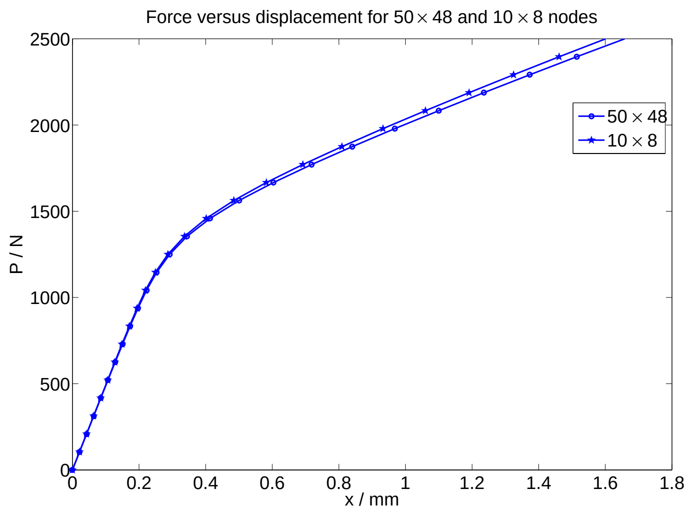

Figure 5.6: Comparison of force versus displacement plot for 10×8 and 50×48 nodes.

gence. A typical convergence serie for a point well inside the plastic region can be seen in Table 5.6.

Iteration	Residual
1	$1.787 \cdot 10^{1}$
2	$1.255 \cdot 10^{-1}$
3	$2.458 \cdot 10^{-6}$
4	$2.028 \cdot 10^{-11}$

Table 5.6: A typical convergence serial.

5.5 Summary

A plasticity model was implemented in a mesh free RKPM program. The implementation was straightforward given a RKPM program for linear statics. The Newton

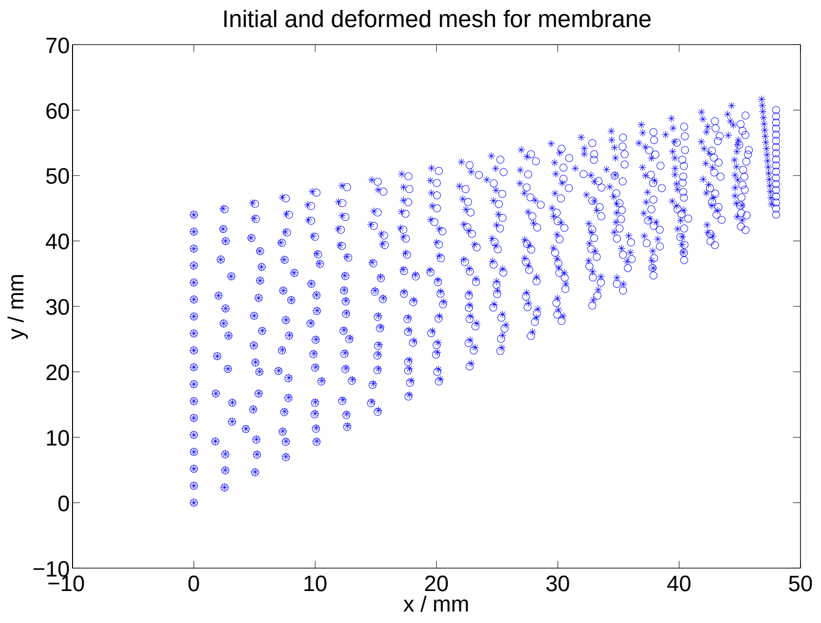

Figure 5.7: A plot of the the inital(o) and deformed(*) mesh for the case of 20 × 18 nodes.

iteration seems to work well and the quadratic convergence is obtained. For the case of small deformation the current configuration is always the initial, therefore we can at the initial part store all shape functions for the integrations points and use them under the whole loading cycle. The test of the membrane and beam shows that even with few number of nodes a accurate results is obtained. If we combine these conclusions we get a fast and accurate program that can be used to solve plasticity problems.

Chapter 6

Large Deformations

6.1 Theory

So far the code has only been tested for material non-linearities as plasticity. The final task now is to apply the code for geometrical non-linearities, i.e. large deformations. The difference compared to the small strain theory is how you choose the strain measure. For small theory we use the so called engineering-strain which is linear with respect to the displacement. For large deformation this relation is not valid any more, so another strain measure and corresponding stress has to be chosen.

6.1.1 Weak form

The new strain measure is the so called Green-Lagrange strain defined by

$$\mathbf{E} = \frac{1}{2}(\mathbf{F}^T\mathbf{F} - \mathbf{I}) = \frac{1}{2}(\mathbf{U} + \mathbf{U}^T + \mathbf{U}^T\mathbf{U}) \tag{6.1}$$

where $\mathbf{F}$ is the deformation gradient, $\mathbf{U}$ the displacement gradient and $\mathbf{I}$ is the identity matrix. It is clear that Green-Lagrange strain contains a extra quadratic term which makes it non linear. The corresponding stress measure is second Piola-Kirchoff stress, $\mathbf{S}$. Second Piola-Kirchoff is related to the Cauchy stress,$\boldsymbol{\sigma}$, as

$$\mathbf{S} = \mathrm{J}\mathbf{F}^{-1}\boldsymbol{\sigma}\mathbf{F}^{-T} \tag{6.2}$$

where

$$\mathrm{J} = \det(\mathbf{F}). \tag{6.3}$$

The weak form in form of the principle of virtual work for a large deformation problem is given from Chen *et al.* (1996) and can be compared with equation 2.15

$$\begin{aligned} -\int_\Omega \delta\mathbf{E}^T\mathbf{S}\, d\Omega + \int_\Omega \delta\mathbf{U}^T\boldsymbol{\phi}^T\mathbf{b}\, d\Omega + \int_{S_t} \delta\mathbf{U}^T\boldsymbol{\phi}^T\mathbf{t}\, dS_t + \\ + \int_{S_u} \delta\boldsymbol{\Lambda}^T\mathbf{N}^T\boldsymbol{\phi}\mathbf{U} - \int_{S_u} \delta\boldsymbol{\Lambda}^T\mathbf{N}^T\overline{\mathbf{u}}\, dS_u + \int_{S_u} \delta\mathbf{U}^T\boldsymbol{\phi}^T\mathbf{N}\boldsymbol{\Lambda}\, dS_u = 0. \end{aligned} \tag{6.4}$$

6.1.2 Introducing approximation

Before proceeding the following two operators are defined

$$\nabla_l = \begin{bmatrix} \frac{\partial}{\partial x} & 0 \\ 0 & \frac{\partial}{\partial y} \\ \frac{\partial}{\partial y} & \frac{\partial}{\partial x} \end{bmatrix} \quad \nabla_u = \begin{bmatrix} \frac{\partial}{\partial x} & 0 \\ \frac{\partial}{\partial y} & 0 \\ 0 & \frac{\partial}{\partial x} \\ 0 & \frac{\partial}{\partial y} \end{bmatrix} \tag{6.5}$$

and a matrix depending on the displacement u

$$A(\mathrm{u}) = \begin{bmatrix} \frac{\partial u_x}{\partial x} & 0 & \frac{\partial u_y}{\partial x} & 0 \\ 0 & \frac{\partial u_x}{\partial y} & 0 & \frac{\partial u_y}{\partial y} \\ \frac{\partial u_x}{\partial y} & \frac{\partial u_x}{\partial x} & \frac{\partial u_y}{\partial y} & \frac{\partial u_y}{\partial x} \end{bmatrix}. \tag{6.6}$$

The Green-Lagrange strain can now be written as

$$\mathbf{E}(\mathbf{x}) = (\mathbf{B}_o + \frac{1}{2}\mathbf{B}_u)\mathbf{U} = \mathbf{BU} \tag{6.7}$$

with the following definitions

$$\mathbf{B}_o = \nabla_l \boldsymbol{\phi} \tag{6.8}$$

$$\mathbf{B}_u = \mathbf{A}(\mathbf{u})\mathbf{H} \quad \text{where} \quad \mathbf{H} = \nabla_u \boldsymbol{\phi}. \tag{6.9}$$

The variation of the strain $\mathbf{E}$ can now with the new approximation be expressed as $\delta\mathbf{E} = \mathbf{B}\delta\mathbf{U}$.

6.1.3 Mesh free formulation

By introducing the notation from section 2.3 and the approximation for the strain it becomes

$$\delta\mathbf{U}^T(\int_\Omega \mathbf{B}^T\mathbf{S}\, d\Omega - \mathbf{f} + \mathbf{G}\boldsymbol{\Lambda}) + \delta\boldsymbol{\Lambda}^T(\mathbf{G}^T\mathbf{U} - \mathbf{q}) = \mathbf{0}. \tag{6.10}$$

With the same arguments as before, the variations are independent and can not always be equal to zero. The following equation system are obtained

$$\boldsymbol{\Psi}(\mathbf{U}, \boldsymbol{\Lambda}) = \left\{ \begin{array}{rcl} \int_\Omega \mathbf{B}^T\mathbf{S}\, d\Omega - \mathbf{f} + \mathbf{G}\boldsymbol{\Lambda} & = & \mathbf{0} \\ \mathbf{G}^T\mathbf{U} - \mathbf{q} & = & \mathbf{0}. \end{array} \right. \tag{6.11}$$

This is the equilibrium equations for a large deformation problem. Because the equations are non-linear, you need to iterate in order to get a solution that fulfills the equilibrium equation. As in section 5.1.1 this is done by a Newton-Raphson algorithm. A linearization around the last known position,$\mathbf{U}_{i-1}$ and $\boldsymbol{\Lambda}_{i-1}$, are done by a Taylor expansion

$$\boldsymbol{\Psi}(\mathbf{U}_{i-1}, \boldsymbol{\Lambda}_{i-1}) + \boldsymbol{\Psi}_{\mathbf{U}}|_{i-1}(\mathbf{U}_i - \mathbf{U}_{i-1}) + \boldsymbol{\Psi}_{\boldsymbol{\Lambda}}|_{i-1}(\boldsymbol{\Lambda}_i - \boldsymbol{\Lambda}_{i-1}) = 0. \tag{6.12}$$

The following derivatives are needed in order to proceed with the calculations. The first derivatives follows from the assumption that the material is linearized, i.e. $d\mathbf{S} = \mathbf{D}d\mathbf{E} = \mathbf{DB}d\mathbf{U}$

$$\int_\Omega \mathbf{B}^T \frac{d\mathbf{S}}{d\mathbf{U}}\, d\Omega = \int_\Omega \mathbf{B}^T\mathbf{DB}\, d\Omega. \tag{6.13}$$

And the second after some matrix manipulations

$$\int_\Omega \frac{d\mathbf{B}^T}{d\mathbf{U}}\mathbf{S}\, d\Omega = \int_\Omega \mathbf{H}^T\mathbf{RH}\, d\Omega \tag{6.14}$$

where

$$\mathbf{R} = \begin{bmatrix} \mathbf{S} & \mathbf{0} \\ \mathbf{0} & \mathbf{S} \end{bmatrix}. \tag{6.15}$$

If we define

$$\mathbf{K} = \int_\Omega \mathbf{B}^T\mathbf{DB}d\Omega + \int_\Omega \mathbf{H}^T\mathbf{RH}\, d\Omega, \tag{6.16}$$

the derivatives needed in equation 6.12 becomes

$$\mathbf{\Psi_U} = \frac{d\mathbf{\Psi}}{d\mathbf{U}} = \begin{cases} \mathbf{K} \\ \mathbf{G}^T \end{cases}$$
$$\mathbf{\Psi_\Lambda} = \frac{d\mathbf{\Psi}}{d\mathbf{\Lambda}} = \begin{cases} \mathbf{G} \\ \mathbf{0}. \end{cases}$$

Again a new variable is introduced

$$\mathbf{a} = \begin{bmatrix} \mathbf{U} \\ \mathbf{\Lambda} \end{bmatrix}. \tag{6.17}$$

So the Taylor expansion, equation 6.12, can in a more compact form be written as

$$\mathbf{\Psi}(\mathbf{a}_i) + \underbrace{\begin{bmatrix} \mathbf{K} & \mathbf{G} \\ \mathbf{G}^T & \mathbf{0} \end{bmatrix}}_{\text{Iteration matrix}}(\mathbf{a}_i - \mathbf{a}_{i-1}) = \mathbf{0} \tag{6.18}$$

Once again we end up with the same iteration format, but this time the stiffness matrix $\mathbf{K}$ takes another form.

Some comments about the results:

- For small deformations $\mathbf{E} = \mathbf{B}_o\mathbf{a}$, now we have a extra non-linear term.
- The extra term not only contains the differentiated shape functions, but also the displacement.
- Note that the expression differs from how the B matrix is constructed, and when you calculate the Green-Lagrange strain.

This excellent matrix formulation is taken from Ristinmaa and Ljung (2002), and it contains a more comprehensive discussion in this subject.

6.1.4 Calculate reaction forces

In previous examples the load has been controlled by a outer force, which has given contribution to the vector **F**. The following examples with large deformation models, the load has to be displacement controlled in order to see the characteristics of the model.

To generate force versus displacement plots, the reaction forces has to be calculated. For that the Lagrange multiplier are used. The upper equation of the equilibrium equation 6.11 contains the term $\mathbf{G}\boldsymbol{\Lambda}$. This term can be interpreted as the term that force the nodes to fulfill the boundary conditions we want to have with help of the Lagrange multiplier. A closer look shows that the term has the same unit as **F**, i.e. Newton. The vector has the dimension number of nodes times one. So the reaction force along a boundary are calculated as a sum of all the elements in $\mathbf{G}\boldsymbol{\Lambda}$ that lies on the boundary.

6.2 RKPM for large deformation with hyperelasticity

Before we reach the final goal with this master thesis, a mesh free program for large deformation with a elasto-plastic material model, a large deformation program for hyperelasticity is implemented. This is just another check to test the accuracy of the program before we reach the goal. There is no extra effort to implement this hyperelastic model, almost all routines are needed for elasto-plastic model.

It is difficult to find easy analytical solutions for large deformation, therefore the RKPM program was tested against the commercial finite element program ABAQUS. A total lagrangian formulation was used, so the shape functions could be calculated in the beginning of the program as before. The energy potential function was a Neo Hooke and given by

$$\varphi = \frac{1}{2}\kappa(J-1)^2 + \frac{1}{2}\mu(J^{-2/3}C_{\alpha\alpha} - 3) \tag{6.19}$$

where $C_{\alpha\beta} = F_{\gamma\alpha}F_{\gamma\beta}$, $J = \det(F_{\alpha\beta})$ and F is the deformation gradient. The second Piola Kirchoff is given by differentiating (6.19) once

$$S_{\alpha\beta} = 2\frac{\partial\varphi}{\partial C_{\alpha\beta}} = \kappa(J^2 - J)C^{-1}_{\alpha\beta} - \frac{1}{3}\mu J^{-2/3}C^{-1}_{\alpha\beta}C_{kk} + \mu J^{-2/3}\delta_{\alpha\beta}. \tag{6.20}$$

To get our constitutive matrix we differentiate (6.19) once again and obtain

$$\begin{aligned} D_{\alpha\beta\gamma\delta} &= 4\frac{\partial^2\varphi}{\partial C_{\alpha\beta}\partial C_{\gamma\delta}} \\ &= 4(\kappa(\frac{J^2}{2}-\frac{J}{4})C^{-1}_{\gamma\delta}C^{-1}_{\alpha\beta}-\frac{\kappa}{4}(J^2-J)(C^{-1}_{\alpha\gamma}C^{-1}_{\beta\delta}+C^{-1}_{\alpha\delta}C^{-1}_{\beta\gamma}) \\ &\quad +\frac{\mu}{18}J^{-2/3}C^{-1}_{\gamma\delta}C_{\alpha\beta}C_{kk}+\frac{\mu}{12}J^{-2/3}(C^{-1}_{\alpha\gamma}C^{-1}_{\beta\delta}+C^{-1}_{\alpha\delta}C^{-1}_{\beta\gamma})C_{kk}) \\ &\quad -\frac{\mu}{6}J^{-2/3}(C^{-1}_{\gamma\delta}\delta_{\alpha\beta}+C^{-1}_{\alpha\beta}\delta_{\gamma\delta})) \qquad (6.21) \end{aligned}$$

6.2.1 Results for beam in extension and Cooks membrane

To check the accuracy of the implementation, the beam test was once again considered. This time simply supported and a constant traction vector in the horizontal direction. Materials parameters was given by $\mu = 3448.276$ MPa and $\kappa = 33333.333$ MPa. The beam was in twelve uniform steps loaded to a total force of 120 kN. For a point (x=48,y=0) the displacement versus force plot is given in Figure 6.1. The same simulation where done in ABAQUS with the same model and material data. A displacement versus load plot for the same point can also be seen in Figure 6.1. The results show excellent agreement between the two methods.

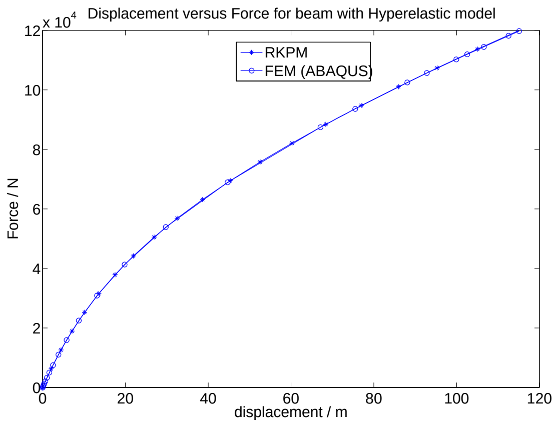

Figure 6.1: Results for beam in extension. Comparison of force versus displacement plot for ABAQUS and RKPM.

But to be even more certain the method was ones again tested for Cooks membrane. Also this time with a constant traction vector in the vertical direction. The structure was loaded up to a total force of -352 kN in 22 equal steps. As the last example it was compared against a finite element solution given by ABAQUS. To be certain that there is not any problem with the element types, a different number of elements was used. For triangular mesh, 3328 elements was generated, and for quadratic mesh, 1740 elements. This compared to a RKPM solution with a total of 1400 nodes is given in Figure 6.2. The figure contains a displacement versus force plot for the different methods.

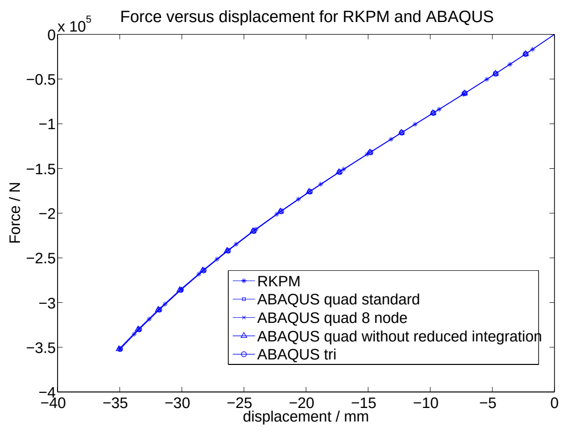

Figure 6.2: Results for Cooks membrane. Comparison of force versus displacement plot for ABAQUS and RKPM.

The results shows excellent agreement also for this slightly more complex example. A plot of the deformed mesh for both RKPM and ABAQUS with triangular elements can be seen in Figure 6.3. The solutions agree well for the entire body, also small details found in the FEM solution can be seen in RKPM. The RKPM method managed to solve large deformation problems for hyperelasticity very well, no further test is needed to check the reliability of the code. The next and final step is now to implement an elasto-plastic model for large deformation theory.

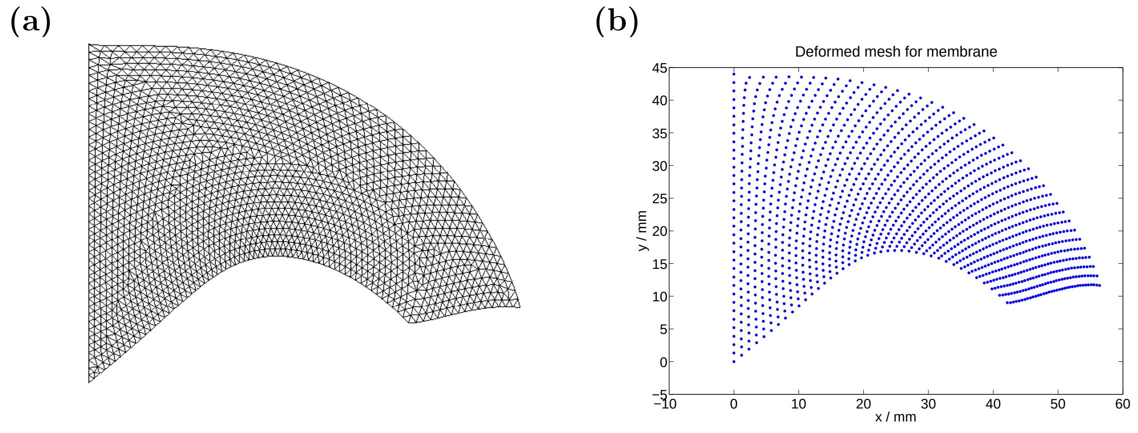

Figure 6.3: A plot of the deformed membrane at the maximum load: a) ABAQUS b) RKPM

6.3 Elasto-plasticity for large deformations

The elasto-plastic model used in the following simulation is an implementation of Box 9.1 in Simo and Hughes (1998) which is a non-linear isotropic hardening model. No further details are discussed in these report, but briefly you can say that the theory is very similar to small deformation, but here we use a multiplicative split of the deformation gradient to an intermediate configuration.

The implementation was easy given the programs implemented before. You need the non-linear geometric part from the hyper-elastic implementation and the variable saving from the plasticity implementation for small deformations, this elasto-plastic model is also path dependent so you need to save variables from last equilibrium. Except that you just have to change the subroutine containing the material model.

The parameters is the same for all examples and is given in Table 6.1. To compare the solution from the mesh free program, the exact same implementation was used as a subroutine to ABAQUS.

$\kappa = 164 \cdot 10^3$ MPa
$\mu = 80 \cdot 10^3$ MPa
$\sigma_{y0} = 450$ MPa
$h = 129.24$ MPa
$\delta = 16.93$
$y_{00} = 265$ MPa

Table 6.1: Material parameters for elasto-plastic model.

6.3.1 Results for beam in extension

The beam example (see section 3.5) was once again considered, this time with a little bit more spectacular effects. To fit better with material data, the geometry was scaled. Before all measures was in meters, now they are in millimeters. The problem was displacement controlled, the right side of the beam was forced in the x-direction. The left side (x=0) was locked in x-direction, and the point (x=0,y=0) was also locked in y-direction. The thickness of the beam was one.

As mentioned before, the exact same simulation was performed in ABAQUS. This example was tested to see how RKPM manage a necking problem. In order to trigger the necking in ABAQUS, a small imperfection was introduced. The left side of the beam was 0.5 millimeters thinner then the right side. As expected the irregular node distribution helped to trigger the necking for the RKPM program, so no imperfection was needed.

The RKPM program was tested in a number of simulation. In order to check how the parameters influence the solution, they was changed in the following manner:

$$\begin{aligned} N_x &= \text{number of nodes in x-direction} \\ N_y &= \text{number of nodes in y-direction} \\ X &= \text{controll the size of support domain as } R_x = X \cdot \frac{L}{N_x - 1},\ R_y = X \cdot \frac{D}{N_y - 1} \\ Y &= \text{number of integrationcells as } \mathsf{nocellsx} = Y \cdot N_x,\ \mathsf{nocellsy} = Y \cdot N_y \end{aligned}$$

Number of integration points in each cell was constant 7×7. For each simulation force versus displacement plot for a node (x=0.048,y=0) was generated. First the number of nodes was constant and the variables X and Y was changed. The results can be seen in Figure 6.4. Then the number of nodes was increased and the same variation was made on X and Y, see Figure 6.5. The results shows big difference when the parameters are changed and no unique solution can be found for this problem.

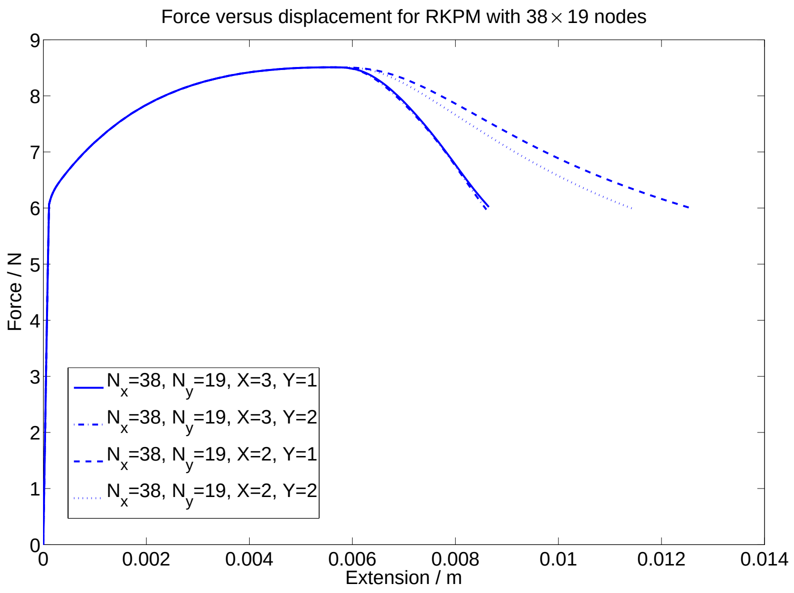

Figure 6.4: A force versus displacement plot of a beam in extension. The nodes are constant 38×19, but number of integration points and size of support domain are changed.

To show the complexity of this problem, a reference solution was made in ABAQUS. For the same problem a force versus displacement plot was generated for different elements. The beam was discretized with approximately 3000 elements. The results can be seen in Figure 6.6, and even ABAQUS have inconsistent results.

To compare the results, one of each test was plotted in Figure 6.7. All methods

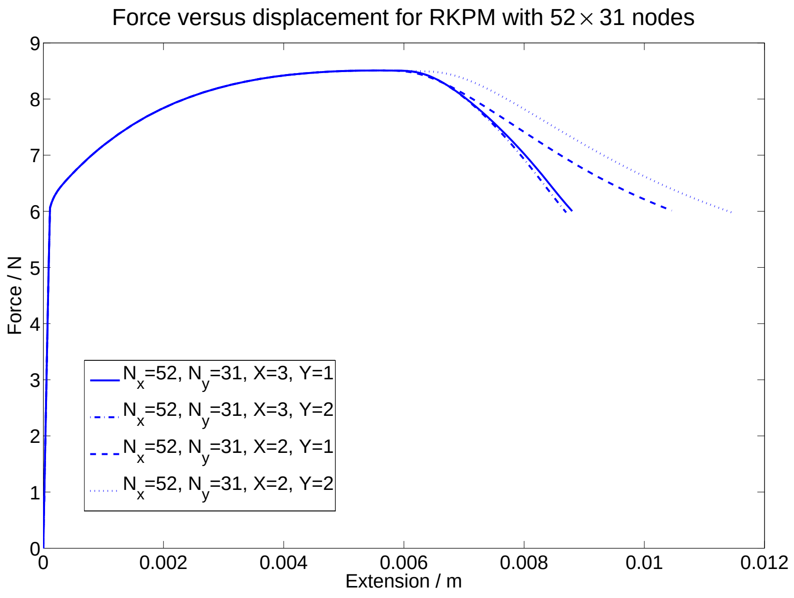

Figure 6.5: A force versus displacement plot of a beam in extension. The nodes are constant 52 × 31, but number of integration points and size of support domain are changed.

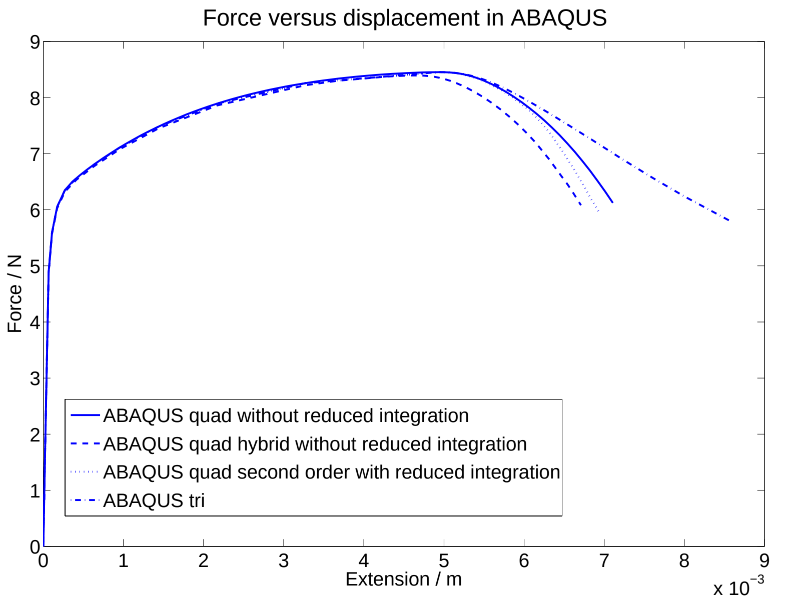

Figure 6.6: A force versus displacement plot of a beam in extension simulated in ABAQUS with different element types. Approximately 6000 elements are used.

and elements give the same results up to the beginning of the necking. The necking starts much earlier in ABAQUS, and for that I have no good answer. If the number of nodes are increased, the necking starts little earlier. If that is a coincident or as a result of more nodes is hard to say.

Something that is hard to see in the force versus displacement plots is that it seems like ABAQUS is softer then RKPM. If the simulation are studied in more detail, it can be seen that in the beginning the necking is the same. Then in most cases for ABAQUS a shear band is developed which results in a very fast necking. This can be seen in Figure 6.8, which contains the deformed beams for both ABAQUS and RKPM at the beginning of necking and at the end of simulations.

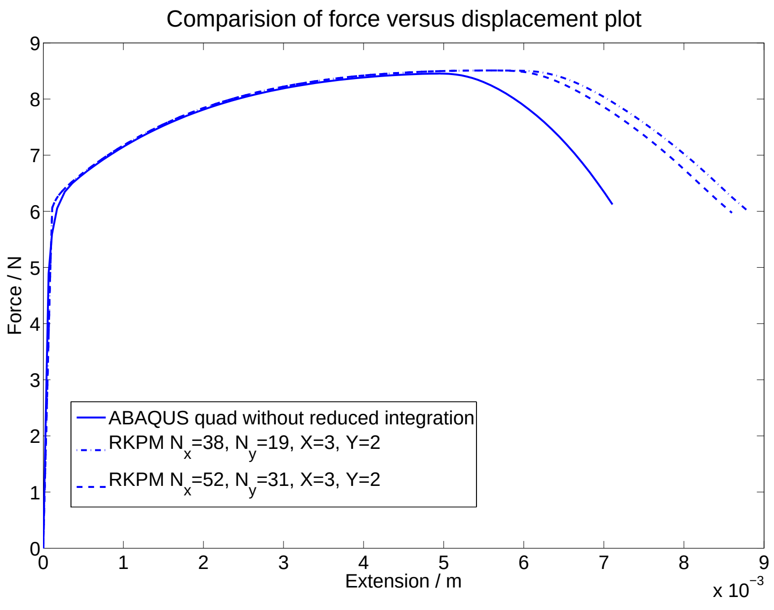

Figure 6.7: Comparision of force versus displacement plot for RKPM and ABAQUS.

6.3.2 Summary

A elasto-plastic model for large deformation has been implemented and tested for a necking example. The solution obtained by RKPM seems to be very sensitive of changes in parameters. But also the commercial program ABAQUS have problems and give different results depending on which elements that is used. The force versus displacement plot have approximately the same appearance except for which time the necking starts.

A interesting observation in this example, the number of integration points does not influence the solution at all. For the example of beam in bending, it was reversed. Integration point have a great influence and size of support domain almost none. This give the conclusion that it is not trivial which parameter should be changed in order to give high accuracy for low cost. If you compare the two parameters with respect to computational cost, it is much cheaper to increase the support domain then the number of integration point.

A comments also has to be made about Figure 6.8. A magnification of the mesh has been done in the end of the necking phase. It should be notice that the mesh should be quadratic. It is not hard to see that this kind of mesh can not give any reasonable results, a remeshing is necessary.

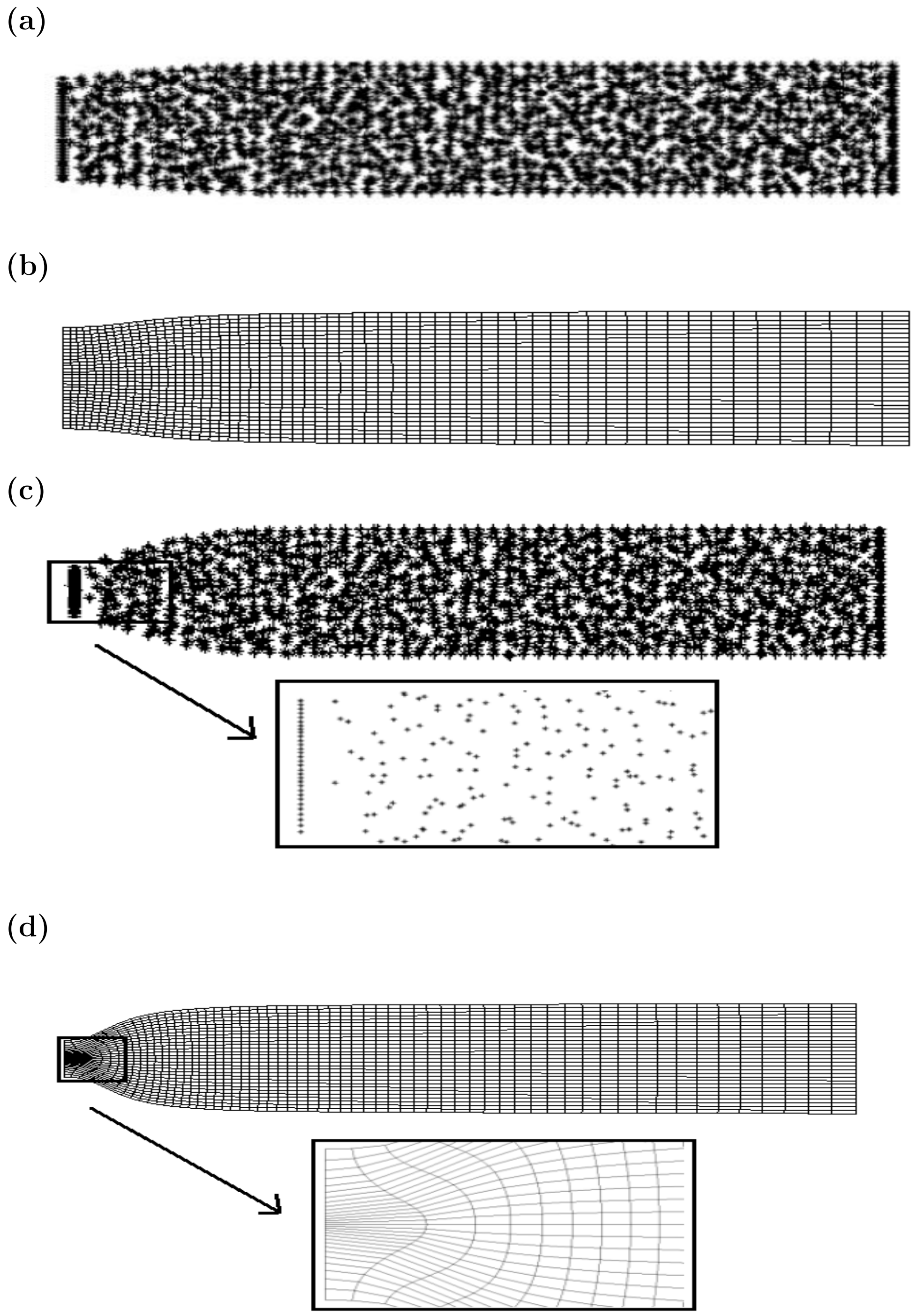

Figure 6.8: A plot of the deformed mesh under necking simulation: a) RKPM, beginning of necking b) FEM, beginning of necking c) RKPM at the end of simulation d) FEM at the end of simulation.

Chapter 7

Conclusions

This master thesis contains an introduction to mesh less methods and their applications. Due to the large material only a overview have been presented and some of the basic methods has been further investigated.

Two of the most known mesh free methods have been implemented and tested, the EFG method and RKPM. The results shows that the difference is small even if the theory in many ways differs. The methods implemented can be considered to be the classic variants, i.e. no modification is made. The last years many articles has been written in order to improve for example

- **Shape functions** so they fulfill Kronecker delta property
- **Integration** in order to avoid the larger number of integration points
- **Coupling with FEM** to speed up calculation
- **Adaptation of support domain** to lower the computational cost

and so on. But none of these new variants have been implemented or discussed in this report. The implementation give accurate result compared to analytical and FEM solutions. But as pointed out in many articles the method needs many integration points, and the solution is sensitive for parameter changes.

As mentioned before the methods were very similar and therefore only RKPM was further developed to large deformations problem. Introducing more complex material models and handling geometric non-linearities is very similar compared to FEM. The results shows that mesh free methods like FEM seems to capture the characteristics that the model try to simulate. For example in cyclic loading with plasticity model or necking of a bar with elasto-plastic model.

Also in large deformation analysis the simplest choice is made concerning which configuration the calculation is made in. For simplicity only a total Lagrange formulation has been used. This have the benefit that the shape function values are the same during the simulation. But this choice feels intuitive strange, to calculate on the initial body when it is highly deformed. But a updated Lagrange formulation would have been much more complicated. For example, should the support domain deform with the body? In that case no search have to be made, because the support domain should contain the same nodes during the simulation. But in my opinion the large deformation would lead to a highly deformed support domain and maybe give rise to same problem as in FEM. But with a constant support domain new search and calculation has to be made through the simulation, and that would not be computational possible. The best solution in my opinion would be a updated Lagrange with constant support domain and the shape functions are updated in a clever way. Another way would be not to calculate new shape functions in every time step, maybe just every twenty or so. In the articles I have read, most seems to use a total Lagrange formulation. Someone used a updated, but with a support domain that deforms with the body.

Concerning improvement of the method, much have all ready been done. But a clever integration method and a easy implemented shape functions that fulfill Kronecker delta property would be preferable. But what I miss most when study all this articles, is an investigation of what impact different parameters have on the solution. Examples of parameters are number of nodes, number of integration points, size of support domain and number of coefficients in basis. In my numerical examples there have been no unified results in this question. For the beam in bending, number of integration points played an significant role. But for the example of necking, number of integration points almost have no influence, instead the support domain had a great impact. So maybe there is not any general answer to this question, but more comprehensive research in this subject would improve the efficiency of the mesh free methods.

Concerning the future of mesh free methods, in my opinion they will never completely replace FEM. But in special applications they have a good chance to be part of the future. The most realistic option is a combination of mesh free and FEM discretization of a body. I do not think it is unrealistic that in a few years when you solve a large deformation problem in a commercial program, you specify the region with large deformation and that region is discretized with only nodes. But before that happen more investigation of mesh free methods have to be made concerning accuracy and stabilization and improvement of especially integration technique.

Bibliography

Atluri, S. and Zhu, T. (1998). A new meshless local petrov-galerkin approach in computional mechanics. *Computional Mechanics*, **22**, 117–127.

Belytschko, T., Lu, Y., and Gu, L. (1994). Element-free galerkin methods. *Int J Number Methods Eng*, **39**, 923–938.

Chen, J.-S., Pan, C., Wu, C.-T., and Liu, W. (1996). Rkpm for large deformation analysis of non-linear structures. *Computer methods in applied mechanics and engineerings*, **139**, 195–227.

Chen, Y., Lee, J., and Eskandarian, A. (2006). *Meshless Methods in Solid Mechanics*. Springer.

Fries, T. and Matthies, H. (2004). Classification and overview of meshfree methods. *Scientific Computing*, **2003-3**.

Gavete, L., Benito, S., Falcon, S., and Ruiz, A. (2000). Implementation of essential boundary conditions in meshless methods. *Communications in numerical methods in engineering*, **16**, 409–421.

Lancaster, P. and Salkauskas, K. (1981). Surfaces generated by moving least squares method. *Mathematic of Computation*, **37**, 141–158.

Liu, G. (2003). *Mesh Free Methods*. CRC Press LLC.

Liu, G. and Liu, M. (2003). *Smoothed Particle Hydrodynamics*. World Scientific.

Liu, W., Jun, S., and Zhang, Y. (1995). Reproducing kernel particle methods. *International Journal for Numerical Methods in Fluids*, **20**, 1081–1106.

Liu, W., Li, S., and Belytschko, T. (1997). Moving least square reproducing kernel methods (i) methodology and convergence. *Computional Methods in Applied Mechanical Engineering*, **143**, 113–154.

Lucy, L. (1977). A numeical approach to the testing of the fission thesis. *Astronom. Journal*, **82**, 1013–1024.

Monaghan, J. (1982). Why particle methods works. *Scientific Computing*, **3**, 422–433.

Most, T. and Bucher, C. (2007). New concepts for moving least squares: an interpolating non-singular weighting function and weighted nodal least squares. *Engineering Analysis with Boundary Elements*.

Nayroles, B., Touzot, G., and Villon, P. (1992). Generalizing the finite element method: diffuse approximation and diffuse elements. *Computional Mechanics*, **10**, 307–318.

Ottosen, N. and Petersson, H. (1992). *Introduction to the finite element method*. Prentice Hall.

Ristinmaa, M. and Ljung, C. (2002). *An introduction to stabilty analysis*. Collected notes.

Ristinmaa, M. and Ottosen, N. (2005). *The Mechanics of Constitutive Modeling*. Elsevier.

Rossi, R. and Alves, M. (2007). On the analysis of an efg method under large deformations and volumetric locking. *Computional Mechanics*, **39**, 381–399.

Simo, J. and Hughes, T. (1998). *Computational Inelasticity*. Springer.

Sukumar, N., Moran, B., and Belytschko, T. (1998). The natural element method in solid mechanics. *International Journal for Numerical Methods in Engineering*, **43**, 839–887.

Timonshenko, S. and Goddier, J. (1970). *Theory of elasticity*. McGraw-Hill.