Institut für Parallele und Verteilte Systeme

Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Masterarbeit Nr. 314159

Development of a FEM code for fluid-structure coupling

Stephan Herb

Studiengang:	Informatik
Prüfer/in:	Prof. Dr. Miriam Mehl
Betreuer/in:	Dipl.-Ing. Florian Lindner
Beginn am:	2015-06-08
Beendet am:	2015-12-08
CR-Nummer:	

Contents

1 Introduction **5**

2 Framework Evaluation **6**
2.1 General Aspects 6
2.2 Frameworks Overview 7
2.2.1 Feel++ 7
2.2.2 OOFEM 7
2.2.3 GetFEM++ 7
2.2.4 MFEM 8
2.2.5 libMesh 8

3 Shell Elements **9**
3.1 Introduction to Linear Elasticity Problems 9
3.2 Plane Element 10
3.2.1 Problem Definition 10
3.2.2 Tri-3 Plane Element 12
3.2.3 Quad-4 Plane Element 19
3.3 Plate Bending Element 23
3.3.1 Problem Definition 23
3.3.2 Tri-3 Plate Element 24
3.3.3 Quad-4 Plate Element 24
3.4 Coordinate Transformation 24
3.5 Shell Element 25

4 FEM Code Implementation **27**
4.1 Introduction to libMesh 27
4.2 libMesh FEM 27
4.3 Parallelization with MPI 28

5 Coupling with preCICE **29**
5.1 Coupling 29
5.1.1 Introduction to coupling 29
5.1.2 Coupling methods 29
5.2 preCICE 29
5.3 Implementation 29

6 Validation **30**
6.1 Test A: Membrane Displacement with Tri-3 30
6.2 Test B: Membrane Displacement with Quad-4 30
6.3 Test C: Plate Displacement with Tri-3 30
6.4 Test D: Plate Displacement with Quad-4 30
6.5 Test E: Shell Displacement with Tri-3 30
6.6 Test F: Shell Displacement with Quad-4 30

6.7 Test G: Convergence (increasing number of elements) 31
6.8 Test H: MPI (increasing number of processes) 31
6.9 Test I: Coupling with preCICE 31

7 Conclusion **32**

1 Introduction

Here comes the introduction. And before that the abstract (that needs to be put into LaTeX as special paragraph)

2 Framework Evaluation

Part of the thesis was to find several frameworks which ease the work with the finite element method. An evaluation of these frameworks was done to select a suitable one for the given task. The evaluation's criteria are presented in this chapter as well as a short description of the studied frameworks.

2.1 General Aspects

In preparation of evaluating the frameworks many criteria were created to objectify the search for the most suitable. The individual aspects were as follows:

- Open Source: All frameworks under consideration need to be published under the GNU Lesser General Public License or similar license that allows modification and/or redistribution.
- Parallelization: In order to accelerate the calculations the framework has to be able to support the widely used Message Passing Interface (MPI).
- The programming language was chosen to be C++. Therefore the framework has to be written in this language.
- Mesh file import: Common mesh files types like gmsh or xda/xdr must be able to load by the framework. Simultaneously the framework must support finite elements like triangles and quadrilaterals with three and four nodes respectively and be able to handle two dimensional elements defined within three dimensional space.
- The framework should handle different types of boundary conditions defined in a mesh file.
- Built-in solvers: In order so solve the matrix-vector-system the framework must provide a variety of different iterative solvers.
- Convenience functions: To optimize the calculations the framework should make use of functions to get matrix-vector and matrix-matrix products, transpose matrix or sparse-matrices.
- Accessible and detailed documentation: In order to guarantee maintainability and expandability the framework has to have a good documentation itself.
- Up-to-date: The framework should be well maintained and actively supported by its developers to ensure a long term compatibility with possible new features of the thesis' code
- The framework should be used by at least a few projects. This shows the framework's importance and usability.
- Easy-to-learn syntax and structure: A rather subjective aspect but an important one. The limited time for the thesis does not allow to study highly complicated structures or semantics. This accompanies the documentation aspect.

2.2 Frameworks Overview

The following list contains FEM libraries and frameworks which were evaluated.

2.2.1 Feel++

- "Feel++ is a unified C++ implementation of Galerkin methods (finite and spectral element methods) in 1D, 2D and 3D to solve partial differential equations."[Con]
- creation of versatile mathematical kernels allow testing and comparing different techniques and methods in solving problems
- focus on close mathematical abstractions regarding partial differential equations (PDE)
- [PCD$^+$12] - imports e.g. gmsh mesh files
- seamlessly parallel with mpi
- currently used in projects at Cemosis (Center for Modeling and Simulation in Strasbourg, France) including fluid structure interactions, high field magnets simulation, or, optical tomography
- actively developed, last major release were on February 2015

2.2.2 OOFEM

[Pat09] - Object Oriented Finite Element Solver (OOFEM) - actively developed with latest release from February 2014
- object oriented architecture; extensible in terms of new element types, boundary conditions or numerical algorithms
- modules for structural mechanics, transport problems and fluid dynamics
- focuses on efficient and robust solution to mechanical, transport and fluid problems
- written in C++ with focus on portability
- interfaces to various external software libraries like PETSc, ParMETIS, or, ParaView
- is used in several publications [Pat]

2.2.3 GetFEM++

- latest release from July 2015 - framework for solving potentially coupled systems of linear and nonlinear PDE
- written in C++ but provides interfaces to languages like Python and Matlab
- model description that gather the variables, data and terms of a problem and some predefined bricks representing classical models
- easy switching from one method to another due to separation of geometric transformation, integration methods, and, finite element method
- can be used to construct generic finite element codes, where methods and the problem's dimension can be changed very easily
- uses MPI for parallelization, though it is stated that "a certain number of procedures are still remaining sequential" [SPR] - imports e.g. gmsh mesh files
- used in project like IceTools [Jar] (open source model for glaciers), EChem++ [BLSS]

(Problem Solving Environment for Electochemistry) and SimNIBS [Thi] (software for Simulation of Non-invasive Brain Stimulation)

2.2.4 MFEM

[mfeb] - The Modular Finite Element Method (MFEM) library acts as a toolbox that provides the building blocks for developing finite element algorithms
- it has a wide range of mesh types, e.g. triangular and quadrilateral 2D elements, curved boundary elements or topologically periodic meshes
- supports MPI-based parallelism throughout the library
- variety of built-in solvers
- written in highly portable C++ and extensible due to separation of mesh, finite element and linear algebra abstractions
- hypre library is tightly integrated within MFEM, for example the use of high-performance preconditioners - The object oriented design of the library as well as the separation of the different parts of the library like the mesh functions, the finite elements, and, the linear algebra, focusing on adapt the code to a variety of applications - use in several publications [mfea]

2.2.5 libMesh

[KPSC06] - actively developed and active user community
- wide variety of mesh file formats to import from (e.g. gmsh, vtk, xda,)
- seamlessly integrated parallel functionality with MPI
- seamlessly interfaces optional external libraries like PETSc or ParMETIS
- complete documentation and documented source code available
- "framework for the numerical simulation of PDE using arbitrary unstructured discretizations on serial and parallel platforms".
- "provide support for adaptive mesh refinement (AMR) computations in parallel"
- supports a variety of 1D, 2D, and 3D geometric and finite element types
- created at The University of Texas at Austin in the CFDLab in March 2002. Contributions have come from developers at the Technische Universität Hamburg-Harburg Institute of Modelling and Computation, CFDLab associates at the PECOS Center at UT-Austin, the Computational Frameworks Group at Idaho National Laboratory, NASA Lyndon B. Johnson Space Center, and MIT.

3 Shell Elements

Mathematical fundamentals of shell elements divided into the two parts and the coordinate transformation

3.1 Introduction to Linear Elasticity Problems

- [Ste15] ch3.1 (S.61)
- Einführung von Vektoren und Matrizen (Verschiebungsvektor, Tensoren bzw. Vektoren der Dehnungen und Spannungen)
- Verknüpfung der Versch. mit den Dehnungen + Stoffgesetz (kinematische Beziehung)
- ???

In the following the fundamental equations of linear elasticity will be considered. Here, the spatial case is used for demonstration, but every lower dimensional problem can easily be derived from it. The following definitions will be used in this thesis:

$$\vec{u}^T = (u\ v\ w) \text{ displacement vector} \tag{1}$$

$$\vec{f}^T = (f_x\ f_y\ f_z) \text{ external force vector} \tag{2}$$

The strains and stresses can either be described in form of tensors $\underline{\epsilon}$ and $\underline{\sigma}$, or as vectors $\vec{\epsilon}$ and $\vec{\sigma}$:

$$\underline{\epsilon} = \begin{pmatrix} \epsilon_{xx} & \epsilon_{xy} & \epsilon_{xz} \\ \epsilon_{yx} & \epsilon_{yy} & \epsilon_{yz} \\ \epsilon_{zx} & \epsilon_{zy} & \epsilon_{zz} \end{pmatrix}; \underline{\sigma} = \begin{pmatrix} \sigma_{xx} & \sigma_{xy} & \sigma_{xz} \\ \sigma_{yx} & \sigma_{yy} & \sigma_{yz} \\ \sigma_{zx} & \sigma_{zy} & \sigma_{zz} \end{pmatrix} \tag{3}$$

$$\vec{\epsilon}^T = \begin{pmatrix} \epsilon_{xx} & \epsilon_{yy} & \epsilon_{zz} & 2\epsilon_{xy} & 2\epsilon_{yz} & 2\epsilon_{zx} \end{pmatrix}; \vec{\sigma}^T = \begin{pmatrix} \sigma_{xx} & \sigma_{yy} & \sigma_{zz} & \sigma_{xy} & \sigma_{yz} & \sigma_{zx} \end{pmatrix} \tag{4}$$

As stated in [Ste15] the relation between displacements and strains is as follows:

$$\underline{\epsilon} = \frac{1}{2}\left(\nabla \vec{u} + \vec{u}\,\nabla\right); \quad \vec{\epsilon} = \underline{L}\vec{u} \tag{5}$$

Equation 5 relates the displacement vector field $\vec{u}$ with the strain field $\underline{\epsilon}$, or $\vec{\epsilon}$ respectively. Here, $\underline{L}$ is a differential operator. This strain-displacement relation is also called *kinematic relationship* [Ste15].
In general initial strains can exist inside the material for example due to temperature changes or shrinkage. Such initial strains are denoted $\vec{\epsilon_0}$ and the stresses will be influenced by the difference between the actual and initial strains. Additionally one could imagine initial residual stresses $\vec{\sigma_0}$ that can be added to the general equation:

$$\vec{\sigma} = \underline{D}\left(\vec{\epsilon} - \vec{\epsilon_0}\right) + \vec{\sigma_0}, \tag{6}$$

where $\underline{D}$ is the material matrix. In the simplest case of linear elasticity with isotropy, $\underline{D}$ only contains two parameters, namely the elastic modulus E (also known as the Young's modulus) and the Poisson's ratio ν. The former one defines the relationship between the stress and strain in a material, the latter one results as the quotient of the fraction of

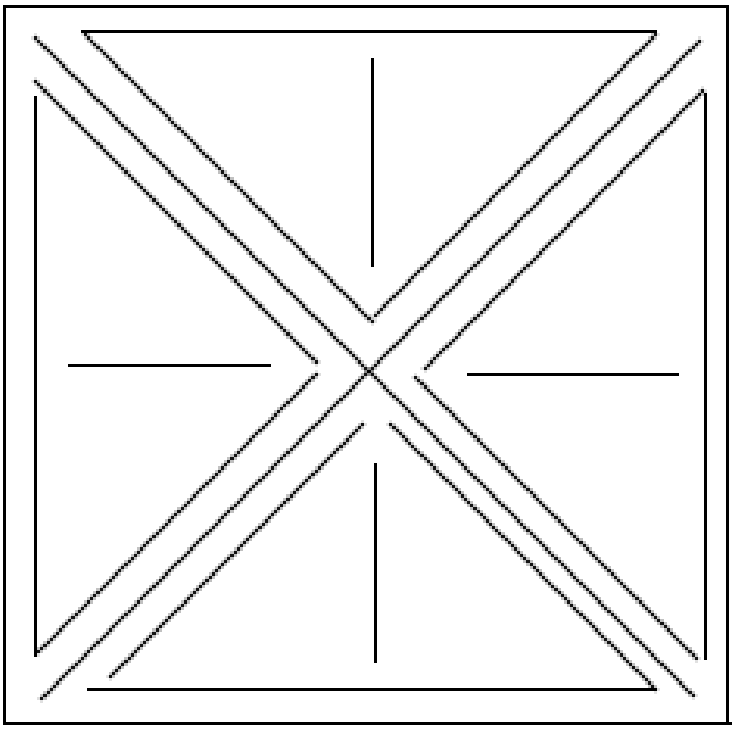

Figure 1: lange Unter-Überschrift

expansion and the fraction of compression for small changes. In the following the initial conditions are ignored, resulting in the a simpler form of equation 6:

$$\vec{\sigma} = \underline{D}\ \vec{\epsilon} \tag{7}$$

For the said isotropic case $\underline{D}$ results in [ZT00]:

$$\underline{D} = \frac{E}{1-\nu^2} \begin{pmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{1-\nu}{2} \end{pmatrix} \tag{8}$$

3.2 Plane Element

First part of shell element: plane part. derivation of this part with two exemplary finite element types

3.2.1 Problem Definition

In figure 5 an object is shown which extends to the x and y axis as its primary direction. The extend in z-direction is smaller and denoted by thickness t. The mid place located

in between the top and bottom surface areas has the coordinate $z = 0$. Its local z-axis equals the normal vector of the mid place. Such an object is called *plane* in the following.

There are two different problem definitions regarding plane elements: Plane stress and plane strain. The directions of displacements u and v along the orthogonal local x and y axis defining its displacement field is a common feature of both problems. Also, both have in common, that only strains and stresses in the xy plane have to be considered: Instead of nine, only three components remain. While in the case of plane stress all other stress components are zero, in plane strain the stress in direction perpendicular to the xy plane is non-zero. In this thesis only plane stress will be discussed in further detail. More information about plane strain is given in [ZT00]. The following conditions must be satisfied such that a plane can be in *plane stress* [Ste15]:

- The thickness t varies only slightly and it must hold: $t/l \ll 1$, with l the extent of the larger side of the plane element.
- The load is applied to the mid place.
- Displacements, strains and stresses are constant across the thickness.

The stress components $\sigma_{xz}, \sigma_{yz}, \sigma_{zz}$ normal to the surface areas with $z \pm t/2$ vanish (equals zero). Therefore only the two normal stress components σ_{xx} and σ_{yy} and the transverse stress component σ_{xy} are left non-zero.

Displacements can only occur in x and y direction. u will be the displacement along x and v along y. The displacement field $\vec{u}$ is as follows:

$$\vec{u} = \begin{pmatrix} u(x,y) & v(x,y) \end{pmatrix}^T \tag{9}$$

The vector for the strain components:

$$\vec{\epsilon} = \begin{pmatrix} \epsilon_{xx} & \epsilon_{yy} & 2\epsilon_{xy} \end{pmatrix}^T \tag{10}$$

Sometimes $2\epsilon_{xy}$ is shortened to γ_{xy} [Ste15]. The vector holding the stress components is similar to that of the strain's vector:

$$\vec{\sigma} = \begin{pmatrix} \sigma_{xx} & \sigma_{yy} & \sigma_{xy} \end{pmatrix}^T \tag{11}$$

The kinematic relationship $\vec{\epsilon} = \underline{L}\vec{u}$ 5 linking the displacements $\vec{u}$ with the strains $\vec{\epsilon}$ at full length:

$$\vec{\epsilon} = \begin{pmatrix} \epsilon_{xx} \\ \epsilon_{yy} \\ 2\epsilon_{xy} \end{pmatrix} = \begin{pmatrix} \frac{\partial u}{\partial x} \\ \frac{\partial v}{\partial y} \\ \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \end{pmatrix} = \begin{pmatrix} \frac{\partial}{\partial x} & 0 \\ 0 & \frac{\partial}{\partial y} \\ \frac{\partial}{\partial y} & \frac{\partial}{\partial x} \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix} = \underline{L}\vec{u} \tag{12}$$

With the strains known and considering equation 5 one can calculate the stresses $\vec{\sigma}$:

$$\vec{\sigma} = \begin{pmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \sigma_{xy} \end{pmatrix} = \frac{E}{1-\nu^2} \begin{pmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{1-\nu}{2} \end{pmatrix} \begin{pmatrix} \epsilon_{xx} \\ \epsilon_{yy} \\ 2\epsilon_{xy} \end{pmatrix} = \underline{D}\vec{\epsilon} = \underline{D}\ \underline{L}\vec{u} \tag{13}$$

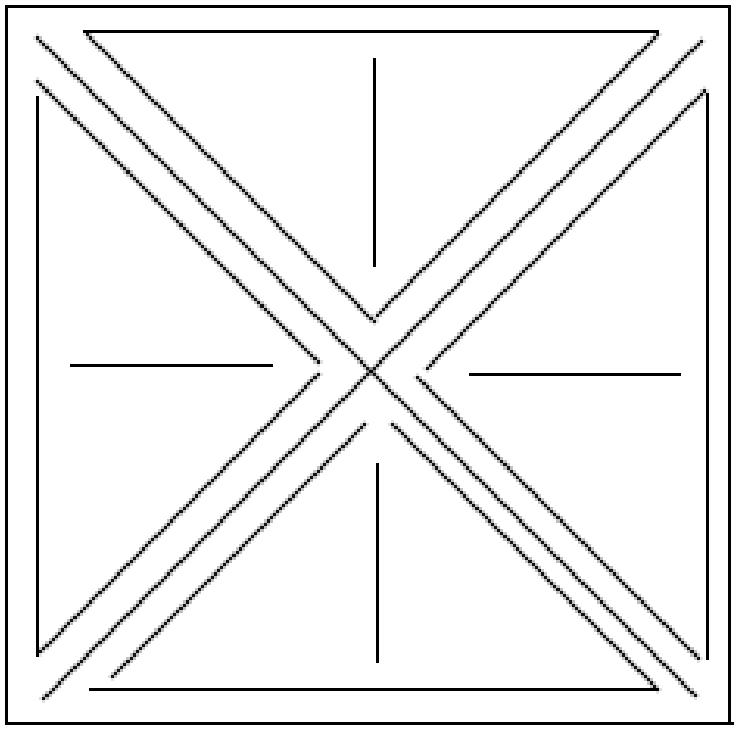

Figure 2: lange Unter-Überschrift

natural boundary conditions in form of nodal forces: $\vec{F} = \begin{pmatrix} F_x & F_y \end{pmatrix}^T$

The total potential of the plane element problem looks as follows:

$$\Pi = 1/2 \int_V \vec{\epsilon}^T \vec{\sigma} dV - \vec{u}^T \vec{F}, \tag{14}$$

with the first term being the elastic strain energy and the second term the single forces.

3.2.2 Tri-3 Plane Element

mathematical derivation of three node triangular plane element **discretization**
see Steinke [Ste15] page 215-221

In section 3.2.1 the plane's functional was derived. Now the focus is on the functional's discretization. Figure 5 shows a general, planar object defined to be placed in the xy-plane. The first discretization step is to divide the object into single triangles approximating the shape of it. This process is called triangulation. Every one of these triangles then represents a finite element with one node at every corner. The finer the triangulation is done the better the object and its boundary are matched by its discrete complement, but also the more finite elements have to be considered in later calculations. One triangular finite element is shown in Figure 5. It is defined by the coordinates

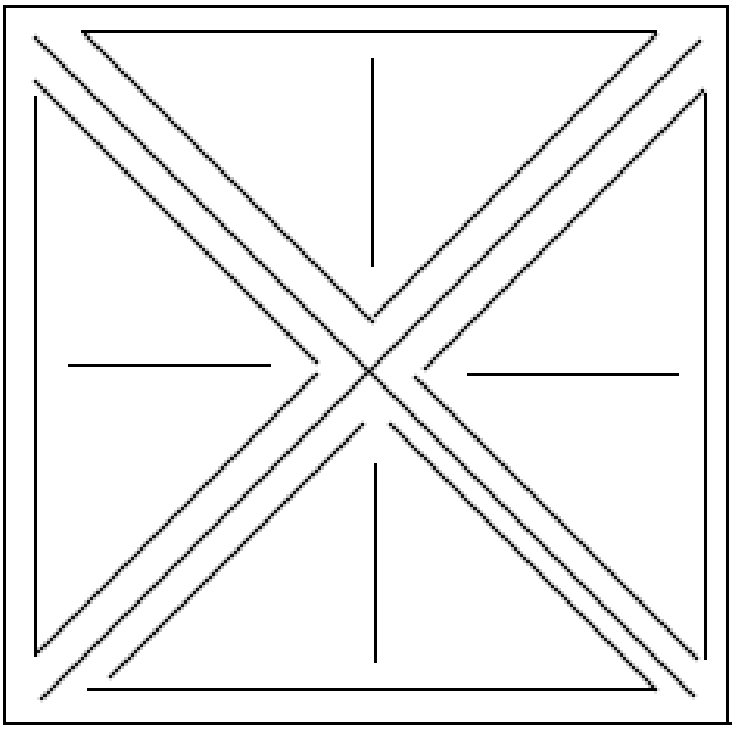

Figure 3: lange Unter-Überschrift

(x_i, y_i) of its three nodes. Since the element is located in the xy-plane, the z-coordinate is of no interest and will be ignored. At every node, forces can be applied denoted with F_{x_i} and F_{y_i}. Accordingly, every node can be displaced. The movement along the x-axis is denoted with u_i, or with v_i along the y-axis respectively. Note, that the node numbering is in counter-clockwise direction. This definition will be kept throughout the thesis, and is important to remember when implementing the FEM-code in order to reduce errors. In this thesis only triangles defined by three nodes are discussed. There are many more finite elements forming triangles, such as six node triangles or even seven node triangles. The main difference between these types of elements are the order of shape functions. More details about higher order triangular finite elements can be found in

In the case of a three node triangle the basis functions for the two displacements u and v are as follows [Ste15]:

$$u(x,y) = a_0 + a_1 L_1 + a_2 L_2 = \begin{pmatrix} 1 & L_1 & L_2 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \end{pmatrix} = \vec{x}^T \vec{a}, \tag{15}$$

$$v(x,y) = a_0 + a_1 L_1 + a_2 L_2 = \begin{pmatrix} 1 & L_1 & L_2 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \end{pmatrix} = \vec{x}^T \vec{a}, \tag{16}$$

both defined in triangular coordinates (see figure 5). To get the unknown coefficients a_i, values for the triangular coordinates are set. This creates a system of linear equations:

$$\begin{aligned} u(L_1 = 1, L_2 = 0) = u_1 \rightarrow u_1 = a_0 + a_1 \\ u(L_1 = 0, L_2 = 1) = u_2 \rightarrow u_2 = a_0 + a_2 \\ u(L_1 = 0, L_2 = 0) = u_3 \rightarrow u_3 = a_0 \end{aligned} \tag{17}$$

Alternatively, this could be also done with v. Written as matrix and vector:

$$\begin{aligned} \underline{A}\vec{a} &= \vec{u} \\ \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \end{pmatrix} &= \begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} \end{aligned} \tag{18}$$

Now, inverting matrix A the coefficients can be found:

$$\vec{a} = \underline{A}^{-1}\vec{u} = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & -1 \\ 0 & 1 & -1 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} \tag{19}$$

If one put equation 19 into 15, or the analogon into 16, the shape functions for the three

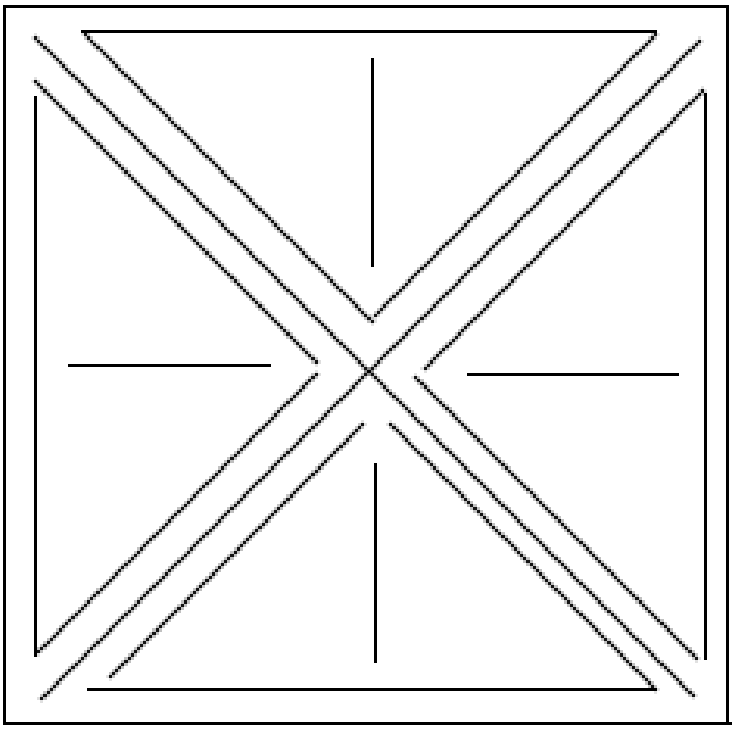

Figure 4: lange Unter-Überschrift

node triangular finite element will be derived, as described in [Ste15]:

$$
\begin{aligned}
u &= \vec{x}^T \vec{a} = \vec{x}^T \underline{A}^{-1} \vec{u} = \vec{N}^T \vec{u} \\
\vec{N}^T &= \vec{x}^T \underline{A}^{-1} = \begin{pmatrix} 1 & L_1 & L_2 \end{pmatrix} \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & -1 \\ 0 & 1 & -1 \end{pmatrix} \\
&= \begin{pmatrix} L_1 & L_2 & 1 - L_1 - L_2 \end{pmatrix} = \begin{pmatrix} N_1 & N_2 & N_3 \end{pmatrix}
\end{aligned} \tag{20}
$$

Characteristically for the shape function, as stated in [Ste15], is, that shape function N_i gets the value 1 at node i and 0 at the two other nodes. The functions are linear with respect to L_1 and L_2 which can be noticed in equation 20. As stated before, these shape functions are the same for displacement u and v. With the knowledge of the displacement values of the element's nodes one can formulate the displacement functions in triangular coordinate notation as follows:

$$
\begin{aligned}
u &= N_1 u_1 + N_2 u_2 + N_3 u_3 \\
v &= N_1 v_1 + N_2 v_2 + N_3 v_3
\end{aligned} \tag{21}
$$

Or in matrix form:

$$
\begin{aligned}
\vec{\tilde{u}} &= \underline{N} \vec{u} \\
\begin{pmatrix} u \\ v \end{pmatrix} &= \begin{pmatrix} N_1 & 0 & N_2 & 0 & N_3 & 0 \\ 0 & N_1 & 0 & N_2 & 0 & N_3 \end{pmatrix} \begin{pmatrix} u_1 \\ v_1 \\ u_2 \\ v_2 \\ u_3 \\ v_3 \end{pmatrix}
\end{aligned} \tag{22}
$$

The vector $\vec{\tilde{u}}$ describes the element's displacements as product of matrix $\underline{N}$ containing the shape functions and vector $\vec{u}$ containing the displacements of the single triangle's nodes. Now, one can put equation 22 into 12:

$$
\vec{\epsilon} = \underline{L} \vec{\tilde{u}} = \underline{L}\, \underline{N} \vec{u} = \underline{B} \vec{u} \tag{23}
$$

The product of $\underline{L}$ and $\underline{N}$ is called *strain-displacement matrix* $\underline{B}$. In order to calculate the strain-displacement matrix, one has to assemble the $\underline{L}$ matrix containing the first partial derivatives of the triangular element. With the chain rule applied, the partial derivatives look as follows:

$$
\begin{aligned}
\frac{\partial}{\partial L_1} &= \frac{\partial x}{\partial L_1} \frac{\partial}{\partial x} + \frac{\partial y}{\partial L_1} \frac{\partial}{\partial y} \\
\frac{\partial}{\partial L_2} &= \frac{\partial x}{\partial L_2} \frac{\partial}{\partial x} + \frac{\partial y}{\partial L_2} \frac{\partial}{\partial y}
\end{aligned} \tag{24}
$$

or in matrix notation:

$$\tilde{\nabla} = \underline{J}\nabla$$
$$\begin{pmatrix} \frac{\partial}{\partial L_1} \\ \frac{\partial}{\partial L_2} \end{pmatrix} = \begin{pmatrix} \frac{\partial x}{\partial L_1} & \frac{\partial y}{\partial L_1} \\ \frac{\partial x}{\partial L_2} & \frac{\partial y}{\partial L_2} \end{pmatrix} \begin{pmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \end{pmatrix}, \tag{25}$$

where $\underline{J}$ represents the Jacobian matrix, ∇ the partial derivatives in Cartesian coordinates and $\tilde{\nabla}$ the partial derivatives in triangular coordinates. To get the derivatives in Cartesian form the upper equation must be multiplied with the inverse Jacobian matrix $\underline{J}^{-1}$:

$$\underline{J}^{-1} = \frac{1}{|\underline{J}|} \begin{pmatrix} \frac{\partial y}{\partial L_2} & -\frac{\partial y}{\partial L_1} \\ \frac{-\partial x}{\partial L_2} & \frac{\partial x}{\partial L_1} \end{pmatrix} \tag{26}$$

The conversion between triangular and Cartesian coordinates can be summarized as follows (see Figure 5 and [Ste15]):

$$\begin{aligned} L_1 + L_2 + L_3 = 1 \rightarrow L_3 &= 1 - L_1 - L_2 \\ x &= x_1 L_1 + x_2 L_2 + x_3 L_3 = (x_1 - x_3)L_1 + (x_2 - x_3)L2 + x_3 \\ y &= y_1 L_1 + y_2 L_2 + y_3 L_3 = (y_1 - y_3)L_1 + (y_2 - y_3)L2 + y_3 \end{aligned} \tag{27}$$

Considering equation 27 the Jacobian matrix can now be calculated:

$$J = \begin{pmatrix} \frac{\partial x}{\partial L_1} = x_1 - x_3 = x_{13} & \frac{\partial y}{\partial L_1} = y_1 - y_3 = y_{13} \\ \frac{\partial x}{\partial L_2} = x_2 - x_3 = x_{23} & \frac{\partial y}{\partial L_2} = y_2 - y_3 = y_{23} \end{pmatrix} = \begin{pmatrix} x_{13} & y_{13} \\ x_{23} & y_{23} \end{pmatrix} \tag{28}$$

and hence the inverse Jacobian matrix:

$$\underline{J}^{-1} = \frac{1}{2A_\triangle} \begin{pmatrix} y_{23} & -y_{13} \\ -x_{23} & x_{13} \end{pmatrix} \tag{29}$$

The determinant of the Jacobian matrix is two times the area of the triangle. With the help of equation 29, 25 can be reorganized

$$\nabla = \underline{J}^{-1}\tilde{\nabla} \tag{30}$$

and this finally yields to the new version of the differential operator $\underline{L}$ [Ste15]:

$$\underline{L} = \frac{1}{2A_\triangle} \begin{pmatrix} y_{23}\frac{\partial}{\partial L_1} - y_{13}\frac{\partial}{\partial L_2} & 0 \\ 0 & -x_{23}\frac{\partial}{\partial L_1} + x_{13}\frac{\partial}{\partial L_2} \\ -x_{23}\frac{\partial}{\partial L_1} + x_{13}\frac{\partial}{\partial L_2} & y_{23}\frac{\partial}{\partial L_1} - y_{13}\frac{\partial}{\partial L_2} \end{pmatrix} \tag{31}$$

Next, the strain-displacement matrix $\underline{B}$ can be calculated:

$$\begin{aligned}
\underline{B} &= \underline{L}\,\underline{N} \\
&= \frac{1}{2A_\triangle}\begin{pmatrix} y_{23}\frac{\partial}{\partial L_1} - y_{13}\frac{\partial}{\partial L_2} & 0 \\ 0 & -x_{23}\frac{\partial}{\partial L_1} + x_{13}\frac{\partial}{\partial L_2} \\ -x_{23}\frac{\partial}{\partial L_1} + x_{13}\frac{\partial}{\partial L_2} & y_{23}\frac{\partial}{\partial L_1} - y_{13}\frac{\partial}{\partial L_2} \end{pmatrix} \\
&\quad \begin{pmatrix} L_1 & 0 & L_2 & 0 & 1-L_1-L_2 & 0 \\ 0 & L_1 & 0 & L_2 & 0 & 1-L_1-L_2 \end{pmatrix} \\
&= \frac{1}{2A_\triangle}\begin{pmatrix} y_{23} & 0 & -y_{13} & 0 & y_{12} & 0 \\ 0 & -x_{23} & 0 & x_{13} & 0 & -x_{12} \\ -x_{23} & y_{23} & x_{13} & -y_{13} & -x_{12} & y_{12} \end{pmatrix}
\end{aligned} \tag{32}$$

With $\underline{B}$ known, one can insert equation 23 into 13 to get the stresses:

$$\vec{\sigma} = \underline{D}\,\underline{B}\vec{u} \tag{33}$$

Finally, every term of the plane element's functional 14 can be filled with the above discretized terms:

$$\begin{aligned}
\Pi &= \frac{1}{2}\int_V \vec{\epsilon}^T\vec{\sigma}\; dV - \vec{u}^T\vec{F} \\
&= \frac{1}{2}\int_V \vec{u}^T\underline{B}^T\underline{D}\;\underline{B}\vec{u}\; dV - \vec{u}^T\vec{F} \\
&= \frac{1}{2}\vec{u}^T\int_V \underline{B}^T\underline{D}\;\underline{B}\; dV\vec{u} - \vec{u}^T\vec{F} \\
&= \frac{1}{2}\vec{u}^T\underline{K}\vec{u} - \vec{u}^T\vec{F}
\end{aligned} \tag{34}$$

with $\underline{K}$ the stiffness matrix and $\vec{F}$ the nodal force vector.

The variation of the functional 34 is as follows [Ste15]:

$$\begin{aligned}
\delta\Pi &= \frac{\partial\Pi}{\partial\vec{u}}\delta\vec{u} = 0 \\
&= \frac{1}{2}\delta\vec{u}^T\frac{\partial\vec{u}^T}{\partial\vec{u}^T}\underline{K}\vec{u} + \frac{1}{2}\vec{u}^T\underline{K}\frac{\partial\vec{u}}{\partial\vec{u}}\delta\vec{u} - \delta\vec{u}\frac{\partial\vec{u}^T}{\partial\vec{u}^T}\vec{F} \\
&= \delta\vec{u}^T\left(\underline{K}\vec{u} - \vec{F}\right) = 0
\end{aligned} \tag{35}$$

In order to satisfy this equation, the term in between the parenthesis must be zero ($\delta\vec{u}^T$ can have arbitrary values). This leads to the equilibrium equation of the triangular plane element as described in [Ste15]:

$$\underline{K}\vec{u} = \vec{F} \tag{36}$$

Since the thickness t of the element is constant per definition, it is $dV = t\; dA$ and therefore the integral of the stiffness matrix changes to:

$$\underline{K} = t\int_A \underline{B}^T\underline{D}\;\underline{B}\; dA = tA_\triangle\underline{B}^T\underline{D}\;\underline{B} \tag{37}$$

3.2.3 Quad-4 Plane Element

Sometimes it can be beneficial to use quadrilateral elements when describing a mesh. In contrast to triangles which always lie, due to their simple shape, in a plane, quadrilateral can have more complex forms. Such cases include for example: The fourth nodes does not lie in the plane defined by the other three or the shape is not convex. It is difficult to deal with such forms and one could be tempted to restrict the element to have rectangular shapes only, because these are easy to formulate and work with. But they are impractical when complicated geometry is to be modeled, especially if details should be emphasized in fine graduation.

One solution to this problem is the use of isoparametric elements. They can be non-rectangular. The trick is to use reference coordinates which map the physical element into a reference element that is a square. Thus, the physical element can have a more general shape, but a coordinate transformation and numerical integration is needed which brings in more mathematical complexity [CMPW02].

In this section quadrilateral isoparametric elements consisting of four nodes are described, but one can expand it to eight or nine node elements. Figure 5 shows the two abstraction layers: On the left side the original element is shown in physical space, on the right side the reference element is shown. The square has a side length of 2. The coordinate system with the ξ and η axis has its origin in the center of the square. Also note the numbering of the nodes is again counter-clockwise.

Similarly to the triangular element, interpolating the displacement field as well as the element geometry is done by shape functions. They are defined in reference coordinates. The displacement of a point within the element can be expressed by the displacements at the nodes and shape functions $\underline{N}$. Also, the position of that point can be expressed in terms of the (global) nodal positions and shape functions $\underline{\tilde{N}}$. The element is called *isoparametric* if $\underline{N}$ is identical to $\underline{\tilde{N}}$. If $\underline{\tilde{N}}$ is of lower degree than $\underline{N}$, the element is called *subparametric* and *superparametric* if it is the other way around [CMPW02].

Every node has two degrees of freedom: A displacement u along the x-axis and a displacement v along the y-axis. To find the shape functions it does not matter which variable to choose, so the following basis function was used for ϕ which can either represent u or v [Ste15]:

$$\phi(\xi,\eta) = a_0 + a_1\xi + a_2\eta + a_3\xi\eta = \begin{pmatrix} 1 & \xi & \eta & \xi\eta \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{pmatrix} = \vec{x}^T\vec{a} \tag{38}$$

The interpolation conditions at the nodes are as follows:

$$\begin{aligned} \phi(-1,-1) = \phi_1 \rightarrow \phi_1 &= a_0 - a_1 - a_2 + a_3 \\ \phi(1,-1) = \phi_2 \rightarrow \phi_2 &= a_0 + a_1 - a_2 - a_3 \\ \phi(1,1) = \phi_3 \rightarrow \phi_3 &= a_0 + a_1 + a_2 + a_3 \\ \phi(-1,1) = \phi_4 \rightarrow \phi_4 &= a_0 - a_1 + a_2 - a_3 \end{aligned} \tag{39}$$

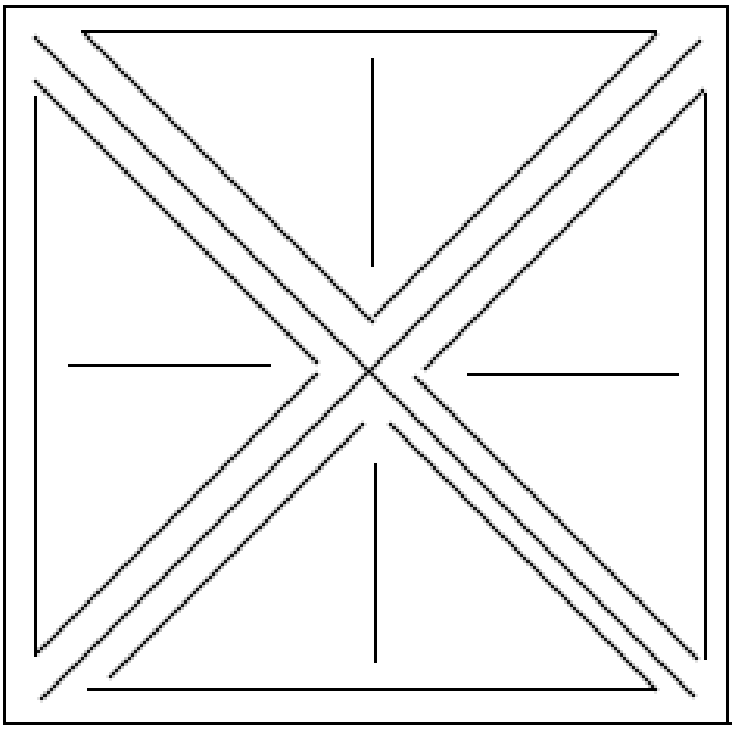

Figure 5: lange Unter-Überschrift

or in matrix notation:

$$\underline{A}\vec{a} = \vec{\phi}$$
$$\begin{pmatrix} 1 & -1 & -1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{pmatrix} = \begin{pmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \\ \phi_4 \end{pmatrix} \tag{40}$$

Inversion of $\underline{A}$ yields the coefficients a_i:

$$\vec{a} = \underline{A}^{-1}\vec{\phi} = \frac{1}{4} \begin{pmatrix} 1 & 1 & 1 & 1 \\ -1 & 1 & 1 & -1 \\ -1 & -1 & 1 & 1 \\ 1 & -1 & 1 & -1 \end{pmatrix} \begin{pmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \\ \phi_4 \end{pmatrix} \tag{41}$$

If the last equation is inserted into 38 one gets the shape functions $\vec{N}$ for the quadrilateral element:

$$\begin{aligned} \phi &= \vec{x}^T \underline{A}^{-1} \vec{\phi} \\ &= \vec{N}^T \vec{\phi} \\ &= \frac{1}{4} \begin{pmatrix} 1 & \xi & \eta & \xi\eta \end{pmatrix} \begin{pmatrix} 1 & 1 & 1 & 1 \\ -1 & 1 & 1 & -1 \\ -1 & -1 & 1 & 1 \\ 1 & -1 & 1 & -1 \end{pmatrix} \vec{\phi} \\ &= \begin{pmatrix} \frac{1}{4}(1-\xi)(1-\eta) & \frac{1}{4}(1+\xi)(1-\eta) & \frac{1}{4}(1+\xi)(1+\eta) & \frac{1}{4}(1-\xi)(1+\eta) \end{pmatrix} \vec{\phi} \end{aligned} \tag{42}$$

To check the correctness of the four shape function one can evaluate shape function i with $\xi\eta$-coordinates of node i. It must evaluate to 1 while at any other node coordinate it must be zero. Now, the displacements can be expressed as follows:

$$\vec{\tilde{u}} = \begin{pmatrix} u \\ v \end{pmatrix} = \underline{N}\vec{u} = \begin{pmatrix} N_1 & 0 & N_2 & 0 & N_3 & 0 & N_4 & 0 \\ 0 & N_1 & 0 & N_2 & 0 & N_3 & 0 & N_4 \end{pmatrix} \begin{pmatrix} u_1 \\ v_1 \\ u_2 \\ v_2 \\ u_3 \\ v_3 \\ u_4 \\ v_4 \end{pmatrix} \tag{43}$$

with $\underline{N}$ being the matrix containing the shape functions and $\vec{u}$ being the vector of the nodal displacements.

The assembly of the strain-displacement matrix $\underline{B}$ is more complicated with isoparametric elements. Due to the $\xi\eta$-coordinates one cannot easily describe an operator such as $\partial/\partial x$. The first step to achieve it, is to formulate a function $\phi = \phi(\xi, \eta)$. Like in the

derivation on the shape functions ϕ can represent u or v. Derivatives with respect to ξ and η are as follows [CMPW02]:

$$\begin{aligned}\frac{\partial\phi}{\partial\xi} &= \frac{\partial\phi}{\partial x}\frac{\partial x}{\partial\xi} + \frac{\partial\phi}{\partial y}\frac{\partial y}{\partial\xi} \\ \frac{\partial\phi}{\partial\eta} &= \frac{\partial\phi}{\partial x}\frac{\partial x}{\partial\eta} + \frac{\partial\phi}{\partial y}\frac{\partial y}{\partial\eta}\end{aligned} \tag{44}$$

or in matrix notation:

$$\vec{\tilde{\phi}} = \underline{J}\vec{\phi} \tag{45}$$

where $\underline{J}$ is the Jacobian matrix:

$$\underline{J} = \begin{pmatrix} \sum N_{i,\xi}x_i & \sum N_{i,\xi}y_i \\ \sum N_{i,\eta}x_i & \sum N_{i,\eta}y_i \end{pmatrix} \tag{46}$$

where $N_{i,j}$ denotes the derivation of the i-th shape function with respect to j and x_i the i-th component of the $\vec{x}$ vector. The Jacobian matrix can be written out as follows:

$$\begin{aligned}\underline{J} &= \frac{1}{4}\begin{pmatrix} -(1-\eta) & (1-\eta) & (1+\eta) & -(1+\eta) \\ -(1-\xi) & -(1+\xi) & (1+\xi) & (1-\xi) \end{pmatrix}\begin{pmatrix} x_1 & y_1 \\ x_2 & y_2 \\ x_3 & y_3 \\ x_4 & y_4 \end{pmatrix} \\ &= \begin{pmatrix} (x_{12}+x_{34})\eta - x_{12} + x_{34} & (y_{12}+y_{34})\eta - x_{12} + y_{34} \\ (x_{12}+x_{34})\xi - x_{13} - x_{24} & (y_{12}+y_{34})\xi - y_{13} + y_{24} \end{pmatrix}\end{aligned} \tag{47}$$

Next, equation 45 can be rearranged to get the derivatives with respect to x and y:

$$\vec{\phi} = \underline{J}^{-1}\vec{\tilde{\phi}} \tag{48}$$

With the derivatives calculated, the strain-displacement relation 5 can be obtained [CMPW02]:

$$\vec{\epsilon} = \begin{pmatrix} \frac{\partial u}{\partial x} \\ \frac{\partial v}{\partial y} \\ \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix}}_{\underline{L}} \begin{pmatrix} \partial u/\partial x \\ \partial u/\partial y \\ \partial v/\partial x \\ \partial v/\partial y \end{pmatrix} \tag{49}$$

$$\begin{pmatrix} \partial u/\partial x \\ \partial u/\partial y \\ \partial v/\partial x \\ \partial v/\partial y \end{pmatrix} = \underbrace{\begin{pmatrix} j_{11} & j_{12} & 0 & 0 \\ j_{21} & j_{22} & 0 & 0 \\ 0 & 0 & j_{11} & j_{12} \\ 0 & 0 & j_{21} & j_{22} \end{pmatrix}}_{\underline{\hat{J}}} \begin{pmatrix} \partial u/\partial\xi \\ \partial u/\partial\eta \\ \partial v/\partial\xi \\ \partial v/\partial\eta \end{pmatrix} \tag{50}$$

$$\begin{pmatrix} \partial u/\partial \xi \\ \partial u/\partial \eta \\ \partial v/\partial \xi \\ \partial v/\partial \eta \end{pmatrix} = \underbrace{\begin{pmatrix} N_{1,\xi} & 0N_{2,\xi} & 0 & N_{3,\xi} & 0 & N_{4,\xi} & 0 \\ N_{1,\eta} & 0N_{2,\eta} & 0 & N_{3,\eta} & 0 & N_{4,\eta} & 0 \\ 0 & N_{1,\xi} & 0N_{2,\xi} & 0 & N_{3,\xi} & 0 & N_{4,\xi} \\ 0 & N_{1,\eta} & 0N_{2,\eta} & 0 & N_{3,\eta} & 0 & N_{4,\eta} \end{pmatrix}}_{\underline{\hat{N}}} \begin{pmatrix} u_1 \\ v_1 \\ u_2 \\ v_2 \\ u_3 \\ v_3 \\ u_4 \\ v_4 \end{pmatrix} \tag{51}$$

$$\underline{B} = \underline{L}\,\underline{\hat{J}}\,\underline{\hat{N}} \tag{52}$$

where j_i denotes the i-th entry in the inverse Jacobian matrix.

Together with the functional equation 34 and the material matrix $\underline{D}$ 13, the stiffness matrix for the quadrilateral isoparametric element can be written as:

$$\underline{K} = \int_V \underline{B}^T \underline{D}\,\underline{B}\,dV = t\int_A \underline{B}^T \underline{D}\,\underline{B}\,dA = t\int_{-1}^{1}\int_{-1}^{1} \underline{B}^T \underline{D}\,\underline{B}\,|\underline{J}|d\xi d\eta \tag{53}$$

For the four node element a Gaussian quadrature needs 2x2 Gauss integration points to satisfy the above equation [Ste15]. These four points are located at $\xi_i = \pm\frac{\sqrt{3}}{3}$ and $\eta_i = \pm\frac{\sqrt{3}}{3}$ with weight factors $\omega_i = 1$. The equation for the stiffness matrix can then be written in discretized form as follows:

$$\underline{K} = t\sum_{i=1}^{2}\sum_{j=1}^{2} \omega_i\omega_j \underline{B}(\xi_i,\eta_j)^T \underline{D}\,\underline{B}(\xi_i,\eta_j)|\underline{J}(\xi_i,\eta_j)| \tag{54}$$

3.3 Plate Bending Element

Second part of shell element: plate part. derivation of this part with two exemplary finite element types

3.3.1 Problem Definition

[Ste15] ch8.1
- bild wie bei **??**
- hier wird die kirchhoff-platten-theorie verwendet
- voraussetzungen der kirchhoff-platte
- größen der platte: durchbiegung w + verdrehungen dw/dx bzw. dw/dy
- dehnungs-verschiebungs-beziehung
- stoffgleichung (krümmungs-momenten-beziehung) -> führt auf Dp und Querkräfte Qx,Qy
- gleichgewichtsbeziehung der platte (p ist flächenlast; näher untersuchen)
- auf verschiedene randbedingungen der platte eingehen
- funktional der platte
- forderungen an plattenelement

3.3.2 Tri-3 Plate Element

[Ste15][Spe88] - mathematical derivation of three node triangular plane element
- see Steinke [Ste15] page 275-282 + [Spe88] for reference of element
- 8.4.4 beschreibt probleme mit einigen dreiecksplattenelementen, deshalb für meine arbeit element von specht [Spe88]
- ansatzfunktion nach specht
- interpolationsbedingungen
- entwicklung der formfunktionen
- krümmungs-verschiebungs-beziehung
- steifigkeitsmatrix

3.3.3 Quad-4 Plate Element

- mathematical derivation of four node quadrilateral plane element
- see Zienkiewicz [ZT00] page 174-177,219-222 [ZTZT77] ??? **[BT82] page 1658-1662**
- formulierung als DKQ element
- **!! formatierung an meine anpassen, da sehr unterschiedlich zu anderen referenzen !!**
- dkq basiert auf diskretisierung der stress-energie, unter vernachlässigung der transverse shear strain energy (gl.(1) s.4)
- krümmungsvektor, Db angeben
- considerations für dkq formulierung
- formfunktionen, koeffizienten
- jacobi-matrix, -inverse, determinante
- krümmungs-verschiebungs-beziehung
- steifigkeitsmatrix

3.4 Coordinate Transformation

erster Entwurf
see [NTRNXB08]; genau: [ZT00]
The nodes and elements in the mesh are defined in global three dimensional space. The elements need to be transformed into local two dimensional space in order to be able to calculate their local stiffness matrix. This local stiffness matrix must then be transformed back into the global system. Transform arbitrary 3D triangle onto xy-Plane:
- given triangle with vertices $A = (a_x, a_y, a_z), B = (b_x, b_y, b_z)$ and $C = (c_x, c_y, c_z)$ ordered in counter-clockwise direction
- let U be the vector from node A to B: $U = B - A = (b_x - a_x, b_y - a_y, b_z - a_z)$ and let V be the vector from node A to C: $V = C - A = (c_x - a_x, c_y - a_y, c_z - a_z)$
- First local unit vector $\tilde{x} = \frac{1}{|U|}U$
- Second local unit vector $\tilde{z} = U \times V \longrightarrow \tilde{z} = \frac{1}{\tilde{z}}\tilde{z}$
- Third local unit vector $\tilde{y} = \tilde{z} \times \tilde{x}$

- Define transformation matrix T as follows:

$$T = \begin{pmatrix} \tilde{x}^T \\ \tilde{y}^T \\ \tilde{z}^T \end{pmatrix} = \begin{pmatrix} \tilde{x}_x & \tilde{x}_y & \tilde{x}_z \\ \tilde{y}_x & \tilde{y}_y & \tilde{y}_z \\ \tilde{z}_x & \tilde{z}_y & \tilde{z}_z \end{pmatrix}$$

- Assembly of element's stiffness needs derivatives. Therefore every triangle can be translated in such a way, that node A lies in the global origin.
- It follows: $\tilde{A} = (0\ 0\ 0)^T, \tilde{B} = \begin{pmatrix} \tilde{b}_x\ 0\ 0 \end{pmatrix}^T, \tilde{C} = (\tilde{c}_x\ \tilde{c}_y\ 0)^T$
- Node A will not be changed by the transformation with T, B will be projected onto the local x-axis due to the definition of it as the vector between A and B and C will be projected onto the local xy-plane.
- One can see that the z component vanishes by transforming into local space

- given quadrilateral with vertices $A = (a_x, a_y, a_z), B = (b_x, b_y, b_z), C = (c_x, c_y, c_z), D = (d_x, d_y, d_z)$ ordered in counter-clockwise direction
- let I be the midpoint of the edge AB as follows: $I = A + \frac{1}{2}(B - A)$. Analogously let J,K and L be the midpoints of the edges BC, CD and DA: $J = B + \frac{1}{2}(C - B), K = C + \frac{1}{2}(D - C), L = D + \frac{1}{2}(A - D)$
- let U be the vector from node L to J: $U = J - L = (j_x - l_x, j_y - l_y, j_z - l_z)$ and let V be the vector from node I to K: $V = K - I = (k_x - i_x, k_y - i_y, k_z - i_z)$
- First local unit vector $\tilde{x} = \frac{1}{\|U|}U$
- Second local unit vector $\tilde{z} = U \times V \longrightarrow \tilde{z} = \frac{1}{|\tilde{z}|}\tilde{z}$
- Third local unit vector $\tilde{y} = \tilde{z} \times \tilde{x}$
- Define transformation matrix T as follows:

$$T = \begin{pmatrix} \tilde{x}^T \\ \tilde{y}^T \\ \tilde{z}^T \end{pmatrix} = \begin{pmatrix} \tilde{x}_x & \tilde{x}_y & \tilde{x}_z \\ \tilde{y}_x & \tilde{y}_y & \tilde{y}_z \\ \tilde{z}_x & \tilde{z}_y & \tilde{z}_z \end{pmatrix}$$

3.5 Shell Element

The combination of the two previous parts and the transformations results in the final shell element
- bild wie bei 8.7 von scheibe und platte und kombination zu schale + erklärungen, welche unbekannten und kräfte man bei welchem teil hat
- erklärung, warum man hier einfach Plane und Plate unabhängig voneinander berechnen und dann zusammenwerfen darf
- gesamtsteifigkeitsmatrix besteht aus blöcken (3x3 bei tri3, 4x4 bei quad4), Ku=F (gleichung 718 bei [Ste15]) - Sei K_m die lokale Steifigkeitsmatrix vom Membran/Plane-Teil und K_p die vom Plate-bending-Teil
- Kij weisst in der Spalte theta_z und Zeile theta_z $(k_i j)_6 6$ eine null auf -> erklärung und warum schlecht. ANDERE REFERENZ ERKLÄRT, WAS WIR DESHALB MACHEN (1/1000 der diagonalwerte)
- Dann muss die (Rück-)Transformationsmatrix T erstellt werden, da SKM im lokalen

KoSys definiert ist, aber in die globale Systemmatrix einsortiert werden muss
- Je nachdem ob 3 oder 4 Knotenelement (Tri-3/Quad-4) sieht K und T natürlich anders aus
- Wir transformieren blockweise von lokal nach global zurück $(K_{ij}, 1 \leq i, j \leq 3(4))$

4 FEM Code Implementation

contains development of the program code with focus on the assembly of the system and its solving, the process of parallelization and the coupling step with preCISE

4.1 Introduction to libMesh

was kann libMesh eigentlich alles; wo unterstützt es einen, was muss man selbst machen

- übersicht über die klassen (im sinne von: für welche probleme kann man es einsetzen)
- noch was allgemeines übersichtmäßiges
- implementierte solver z.b. für elasticity problem; hier in doxygen function description reinschauen
- funktionen wie assembly_matrix muss user selbst füllen
- boundary_info mit ids usw. automatisch erstellt bei mesh import
- boundary conditions erstellbar -> automatisch constrains system und rhs

4.2 libMesh FEM

details about the implementation with the libmesh FEM framework:
- initialization: loading of parameters, setting up libmesh (evtl. uninteressant und es kann weg, oder es muss noch mehr hier rein)
- mesh loading/import: wie sieht mesh file aus bzw. welche typen werden akzeptiert, welche ids für bcs müssen verwendet werden
- set up of system: erstellen des linearimplicitsystems, erstellen der variablen, der bcs, des solvers usw.
- assembly of system matrix and RHS: größter teil; hier wird auf die erstellung der lokalen und globalen stiffnessmatrix eingegangen (integral mit gauss-quadratur lösen z.B. [Ste15] s.248), das auslesen der forces und der entsprechende eintrag in der rhs gesetzt; das mitverfolgen der bereits bearbeiteten knoten mittels unordered_set usw.
- boundary conditions: eventuell bereits unter setup; grundsätzlich auf die beiden bc-typen eingehen, wie das in libmesh gelöst wurde
- solving and getting the result vector: das lösen an sich ist eine code-zeile. hier kann man aber schreiben, mit was libmesh umgehen kann an lösern, welche einstellmöglichkeiten es gibt (error-eps, #iters). und es geht drum, wie man an die tatsächlichen werte für die displacements kommt und was daraus am ende wird
- für die standalone-version noch ein absatz zur ausgabe in exodus2-file

4.3 Parallelization with MPI

additional steps to make the code ready for multi process execution with MPI
- viel ist es nicht, was man tun muss, damit libmesh mit mpi läuft
- grundsätzlich ist zum lösen des gleichungssystem mit mehreren prozessen petsc als externe lib notwendig
- am mesh muss nichts verändert werden, da libmesh automatisch eine partitionierung des meshes vornimmt (kann aber verbessert werden)
- damit rhs korrekt gesetzt wird muss über die prozessgrenzen hinweg klar sein, ob knoten bereits bearbeitet wurde oder nicht. wie das gelöst wurde kommt hier rein

5 Coupling with preCICE

still under construction; just a rough idea what this section will contain

5.1 Coupling

short introduction what coupling means (in this case)

5.1.1 Introduction to coupling

todo: ref preCICE phd, andere ref evtl.

5.1.2 Coupling methods

...

5.2 preCICE

short introduction what preCICE is

5.3 Implementation

"modification" of the code to work with preCICE

6 Validation

The code was tested with several problems to validate its correctness and state where and why there are differing results to the existing (commercial) FEM codes

6.1 Test A: Membrane Displacement with Tri-3

Sprungbrett bestehend aus 8 Elementen; links fest eingespannt, rechts Kraft in y-Richtung an beiden Randknoten
Sprungbrett bestehend aus 32 Elementen in Fischgrätmuster angeordnet; selbe BCs aber andere Kraftwerte
test_c.xda, test_d.xda - beides korrekt

6.2 Test B: Membrane Displacement with Quad-4

Sprungbrett bestehend aus 3 Elementen mit BC wie in Test A aber einzelne Kraft auf oberen rechten Knoten in neg. y-Richtung
selbes Mesh wie in test_d.xda nur eben mit 16 Elementen. Selbe BCs, selbe Kraftwerte
test_e.xda, test_f.xda - beides korrekt

6.3 Test C: Plate Displacement with Tri-3

Platte an allen 4 Seiten eingespannt. Einzelne Kraft im Zentrum in neg. z-Richtung
Alternativ mit anderen Parametern test_g
test_a_triN.xda, test_g_triAB_N.xda - korrekt, noch nicht getestet

6.4 Test D: Plate Displacement with Quad-4

Selbes mesh wie Test C nur eben mit Quadelementen
test_a_quadN.xda, test_g_quad_N.xda - korrekt, noch nicht getestet

6.5 Test E: Shell Displacement with Tri-3

Ein H-Trägerbalken. Am einen Ende fest eingespannt. Am anderen Ende wird oben eine Kraft am äußeren Knoten in den Balken hinein in flacher Ebene gegeben, gleichzeitig wird unten an der gegenüberliegenden Seite eine Kraft in entgegengesetzter Richtung gegeben
test_j_tri.xda - korrekt

6.6 Test F: Shell Displacement with Quad-4

Gleich wie Test E nur eben Quadelemente
test_j_quad.xda - korrekt

6.7 Test G: Convergence (increasing number of elements)

??? theoretisch mit Test C/D bereits durchführbar mit N=2,4,8,16,32,64,128

6.8 Test H: MPI (increasing number of processes)

??? theoretisch alle Tests, z.B. E/F mit Prozessoranzahl = 1,2,4,8,16. In dem Fall ist natürlich die Zeit interessant und ob die Ergebnisse jeweils alle gleich sind

6.9 Test I: Coupling with preCICE

???

7 Conclusion

What does my code do, what problems arose, what problems persist, what does my code cannot do, where are opportunities for extensions, etc.

References

[BLSS] Martin Bogdan, Kai Ludwig, Elena Sapozhnikova, and Bernd Speiser. Echem++ - a problem solving environment for electrochemistry. `http://www.echem.uni-tuebingen.de/echem/software/EChem++/echem++.shtml/`.

[BT82] Jean-Louis Batoz and Mabrouk Ben Tahar. Evaluation of a new quadrilateral thin plate bending element. *International Journal for Numerical Methods in Engineering*, 18(11):1655–1677, 1982.

[CMPW02] Robert D. Cook, David S. Malkus, Michael E. Plesha, and Robert J. Witt. *Concepts and Applications of Finite Element Analysis*. J. Wiley, 2002.

[Con] Feel++ Consortium. Feel++ - finite element embedded library in c++. `http://www.feelpp.org`.

[Jar] Alexander H. Jarosch. icetools: a numerical ice flow model download. `http://sourceforge.net/projects/icetools/`.

[KPSC06] B. S. Kirk, J. W. Peterson, R. H. Stogner, and G. F. Carey. `libMesh`: A C++ Library for Parallel Adaptive Mesh Refinement/Coarsening Simulations. *Engineering with Computers*, 22(3–4):237–254, 2006. `http://dx.doi.org/10.1007/s00366-006-0049-3`.

[mfea] Mfem - finite element discretization library. `http://mfem.org/publications/`.

[mfeb] MFEM: Modular finite element methods. `http://www.mfem.org`.

[NTRNXB08] N Nguyen-Thanh, Timon Rabczuk, H Nguyen-Xuan, and Stéphane PA Bordas. A smoothed finite element method for shell analysis. *Computer Methods in Applied Mechanics and Engineering*, 198(2):165–177, 2008.

[Pat] Bořek Patzák. publications [oofem wiki]. `http://www.oofem.org/wiki/doku.php?id=publications`.

[Pat09] Bořek Patzák. Oofem project home page, 2009. `http://www.oofem.org`.

[PCD+12] Christophe Prud'Homme, Vincent Chabannes, Vincent Doyeux, Mourad Ismail, Abdoulaye Samaké, and Gonçalo Pena. Feel++: A computational framework for galerkin methods and advanced numerical methods. In *ESAIM: Proceedings*, volume 38, pages 429–455. EDP Sciences, 2012.

[Spe88] Bernhard Specht. Modified shape functions for the three-node plate bending element passing the patch test. *International Journal for Numerical Methods in Engineering*, 26(3):705–715, 1988.

[SPR] Luis Saavedra, Julien Pommier, and Yves Renard. Mpi parallelization of getfem++ - getfem++. http://download.gna.org/getfem/html/homepage/userdoc/parallel.html.

[Ste15] Peter Steinke. *Finite-Elemente-Methode: Rechnergestützte Einführung*. Springer-Verlag, 2015.

[Thi] Axel Thielscher. free software package for the simulation of non-invasive brain stimulation. http://simnibs.de/.

[ZT00] Olgierd Cecil Zienkiewicz and Robert Leroy Taylor. *The finite element method: Solid mechanics*, volume 2. Butterworth-heinemann, 2000.

[ZTZT77] Olgierd Cecil Zienkiewicz, Robert Leroy Taylor, Olgierd Cecil Zienkiewicz, and Robert Lee Taylor. *The finite element method*, volume 3. McGraw-hill London, 1977.

Erklärung

Ich versichere, diese Arbeit selbstständig verfasst zu haben. Ich habe keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommene Aussagen als solche gekennzeichnet. Weder diese Arbeit noch wesentliche Teile daraus waren bisher Gegenstand eines anderen Prüfungsverfahrens. Ich habe diese Arbeit bisher weder teilweise noch vollständig veröffentlicht. Das elektronische Exemplar stimmt mit allen eingereichten Exemplaren überein.

Ort, Datum, Unterschrift