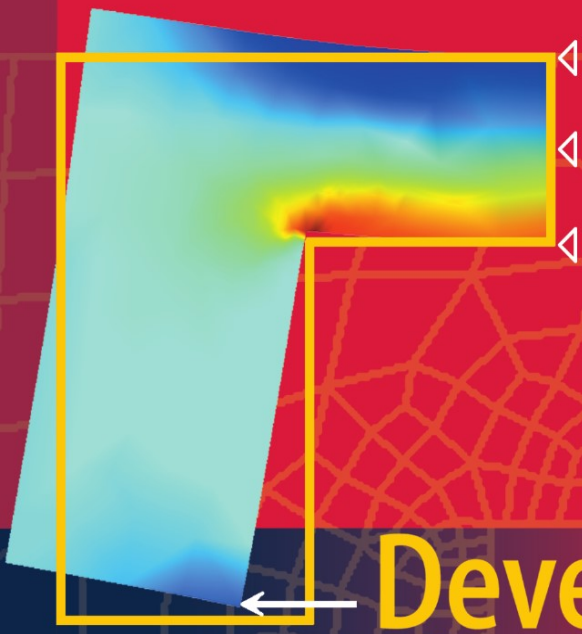

Herbert Baaser

Development and Application of the Finite Element Method Based on MatLab

Springer

Development and Application of the Finite Element Method based on MATLAB

Herbert Baaser

Development and Application of the Finite Element Method based on MATLAB

Springer

Dr.-Ing. Herbert Baaser
Private Lecturer at
Darmstadt University of Technology – FB 13/Solid Mechanics
Hochschulstr. 1
64289 Darmstadt, Germany

and

Senior Scientist at
Freudenberg R&D Services
69465 Weinheim, Germany
E-mail: DAEdalon@Baaserweb.de

ISBN 978-3-642-13152-3 e-ISBN 978-3-642-13153-0

DOI 10.1007/978-3-642-13153-0

Library of Congress Control Number: 2010926861

Typesetting: Data supplied by the authors

Production & Cover Design: Scientific Publishing Services Pvt. Ltd., Chennai, India

Printed on acid-free paper

9 8 7 6 5 4 3 2 1

springer.com

To my family
with respect, thanks and honor.

Preface

The intention of this booklet is a brief but general introduction into the treatment of the **F**inite **E**lement **M**ethod (FEM). The FEM has become the leading method in computer–oriented mechanics, so that many scientific branches have grown up besides over the last decades. Nevertheless, the FEM today is a question of economy. On the one hand its industrial application is forced to reduce product development costs and time, on the other hand a large number of commercial FEM codes and a still growing number of software for effective pre– and postprocessors are available in the meantime.

Due to that, today it is a quite challenging task to operate with all these different tools at the same time and to understand all handling and solution techniques developed over the last years. So, we want to help in getting a deeper insight into the main "interfaces" between the "customers of the FEM" and the codes itself by providing a totally open structured FE–code based on MATLAB, which is a very powerful tool in operating with matrix based formulations. That idea and conditions forced us some years ago to initiate `DAEdalon` as a tool for general FE developments in research applications. In spite of still existing high sophisticated – mostly commercial – FE codes, the success and the acceptance of such a structured tool justify that decision afterwards more and more.

In the main focus are students of applied mechanics, mechanical and civil engineering sciences and interested CAx engineers doing their daily job and want to know what happens behind. For all of them, this book enables a self–study in doing and implementation of given or self developed algorithm within the wide field of the Finite Element Method. Thus, one can easily go into a training in basics of element and material formulations.

Additionally, some sections or hints may animate the reader for a critical reflexion of his doing in using commercial FE codes and relating applications. Often, this enables an increase of effectivity in the state of development when there is a open tool for implementations before one goes to the user interfaces of commercial FE systems. By that understanding, the FEM can also be seen as optimal approximation method for nonlinear (solid) continuum mechanics

as well for the spatial description as for the nonlinearities of the material law, which become a more and more relevance in engineering simulations.

Especially, this guide is no comprehensive study of the FEM in general like the great schools summed up in e.g. Hughes [2000], Stein et al. [2004] or Wriggers [2001]. Nor this is a new treatise of computational inelasticity like Simo & Hughes [1998] or of nonlinear continuum mechanics as Holzapfel [2000]. All of them influenced me without their explicit knowing, for which I am very thankful with high respect. Nevertheless, the reading of these books, the working with the FEM and the teaching of nonlinear methods to many interested graduated and PhD students for several years motivate me again and again to filter out some details with strong focus to the numerical application.

To this end, we invite the interested reader of this book to get in contact with `DAEdalon` by visiting the homepage URL `http://www.DAEdalon.org`. Here, the described open FEM system can be obtained completely in form of `m-files` for the Matlab environment. Additionally, we give some example input files cited in this booklet at `www.DAEdalon.org/Examples`. Further on, we provide an e–mail contact for questions, hints and also critical remarks in order to improve the `DAEdalon` package following the requests of the user community.

Finally, I send special thanks to my colleagues and respectable friends Steffen Eckert, Volkmar Mehling, Oliver Goy, Amin Mozaffarin and Ralf Müller at Darmstadt University of Technology for many discussions and still ongoing code development of essential parts of `DAEdalon`.

Bingen–Büdesheim, Germany Herbert Baaser
Summer 2010

Contents

1

A Quick Start into DAEdalon

1.1 Download and Installation

The latest and stable version of DAEdalon is provided for download at the URL http://www.DAEdalon.org in order to obtain a tar–file, which can be unpacked with any usual archive program.

Doing so, one gets a folder and file structure with ../DAEdalon_v2.00 as kernel followed by a substructure which is necessary for a run of the system. Please set your MATLAB working directory to that path DAEdalon_v2.00 by the selection function in your MATLAB system[1].

In the following, DAEdalon assumes to read the current input data out of the folder ../DAEdalon_v2.00/input, just one branch below the actual working directory.

1.2 Elastomechanical Example

Let's start with some considerations on a simple elastomechanical example as given in Fig. 1.1 in order to get a first insight into the basics and functionalities of the **F**inite **E**lement **M**ethod and especially the open structure of DAEdalon.

To restrict ourselves, we look onto a thin, linear elastic structure, fixed at a part of its boundary and loaded by a concentrated load **F**. In that case, the material behavior is given by two parameters, the YOUNG modulus as $E = 2.1 \cdot 10^5$ MPa and the POISSON ratio as $\nu = 0.3$, respectively.

We are interested in the deflection of the Panel and the stress distribution inside.

[1] Hint: Search for *directory* in the MATLAB help mode.

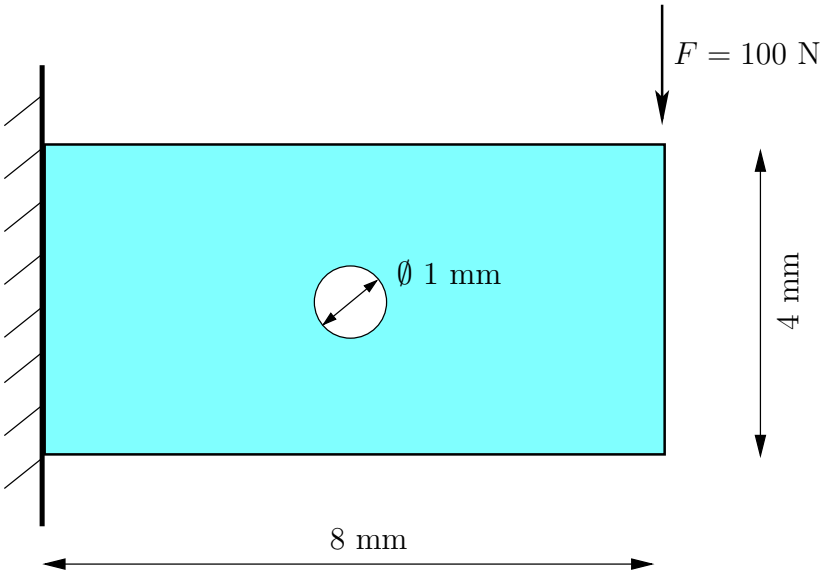

Fig. 1.1 Example of a Thin Panel with a Hole

1.3 Principle of Virtual Displacements

In order to get a short introduction into DAEdalon and the mechanical *story behind*, we write down the weak form of equilibrium as initial boundary value problem of linear elasto–static

$$\int_v \boldsymbol{\sigma} : \delta\boldsymbol{\varepsilon}\, \mathrm{d}v = \int_a \mathbf{t}^{\mathrm{T}} \cdot \delta\mathbf{u}\, \mathrm{d}a \ , \tag{1.1}$$

which we discretize and solve by DAEdalon within the MATLAB environment. In (1.1) the left hand side describes the virtual internal work and the term of the right hand side represents the virtual work of the surface loads $\mathbf{t}$ on the considered structure — neglecting possible body forces without restricting generality.

Firstly, we consider a two–dimensional (2d) case with plane stress assumption ($\sigma_{33} \equiv 0$) in a small strain regime, where we notice the coefficients of the symmetric stress tensor $\boldsymbol{\sigma}$ in vector type form exploiting the so called VOIGT notation as

$$\check{\boldsymbol{\sigma}} = \begin{bmatrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{12} \end{bmatrix} = \begin{bmatrix} \sigma_1 \\ \sigma_2 \\ \sigma_4 \end{bmatrix} \ . \tag{1.2}$$

Doing that in an analogue manner with the strain measure $\boldsymbol{\varepsilon}$ except of the shear entries, which we notice as double

$$\check{\boldsymbol{\varepsilon}} = \begin{bmatrix} \varepsilon_{11} \\ \varepsilon_{22} \\ 2\,\varepsilon_{12} \end{bmatrix} = \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_4 \end{bmatrix} \ , \tag{1.3}$$

we are able to write (1.1) as

$$\int_v \delta\check{\boldsymbol{\varepsilon}}^{\mathrm{T}} \cdot \check{\boldsymbol{\sigma}} \, \mathrm{d}v = \int_a \delta\mathbf{u}^{\mathrm{T}} \cdot \mathbf{t} \, \mathrm{d}a \ . \tag{1.4}$$

Please note, that we are able to formulate (1.4) in such a manner by using (1.2) and (1.3) in order to compute the double–contracting product in (1.1) by the scalar product $\delta\check{\boldsymbol{\varepsilon}}^{\mathrm{T}} \cdot \check{\boldsymbol{\sigma}} \equiv \boldsymbol{\sigma} : \delta\boldsymbol{\varepsilon}$, where we used the symmetry of $\boldsymbol{\sigma}$ and $\boldsymbol{\varepsilon}$ and especially the notation in (1.3) to obtain the equivalence.

As constitutive model, we use in that prologue HOOKEs law, which connects in the simplest way the strains and the stresses by a fourth order tensor $\check{\mathbb{C}}$ given by YOUNGs modulus E and the POISSON ratio ν. Making use of the above described VOIGT notation, we can write

$$\check{\boldsymbol{\sigma}} = \check{\mathbb{C}} \cdot \check{\boldsymbol{\varepsilon}} = \frac{E}{1-\nu^2} \begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{1-\nu}{2} \end{bmatrix} \cdot \check{\boldsymbol{\varepsilon}} \tag{1.5}$$

for a plane stress formulation.

1.4 Discretization of Basic Equations

We are now in the situation to discretize the given problem in form of (1.4). Let us assume to approximate as well the geometry $\mathbf{x}$ as the displacements $\mathbf{u}$ by the same functions N. This proceeding is known as *isoparametric concept*. In parallel, we have to subdivide the considered structure into smaller parts — the finite elements, as can exemplary be seen in Fig. 1.2. These finite elements assemble our structure spatially. They are declared geometrically by the position of the nodes. From the mathematical point of view, the nodes are defined by their discrete positions $\mathbf{x}_I = [x\,y]_I^{\mathrm{T}}$ and *carry* the mechanical fields likewise in discrete form. In our example, the displacements field $\mathbf{u}$ is of interest and it is the unknown field in (1.1) and (1.4). So, that quantity exists due to the discretization on the nodes as $\mathbf{u}_I = [u_x\,u_y]_I^{\mathrm{T}}$, where I counts the number of nodes in the problem. Usually, one concentrate the look onto the element level and give the above mentioned approximation by counting over the element nodes — in that case, we choose a discretization and approximation by classical 4–noded elements (often called *quadrilaterals*). The geometry and the displacement field is given element–wise by

$$\mathbf{x}^e = \sum_{I=1}^{4} N_I(.)\,\mathbf{x}_I \quad \text{and} \quad \mathbf{u}^e = \sum_{I=1}^{4} N_I(.)\,\mathbf{u}_I \ . \tag{1.6}$$

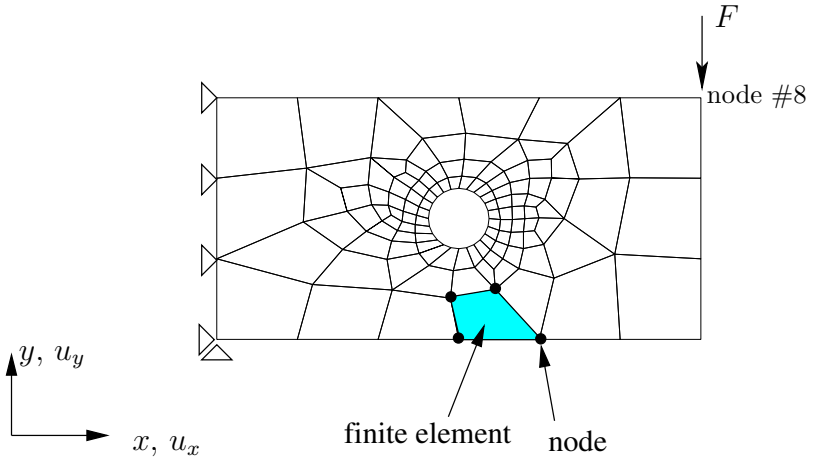

Fig. 1.2 Discretized Structure with Load **F** and Boundary Conditions

More details about the so called *shape functions N* should be skipped in that section and will be picked up in Sec. 2.6.2. Nevertheless, now we are able to give the discretization of the strain field $\boldsymbol{\varepsilon}$ as spatial derivative of the discrete displacement field as

$$\boldsymbol{\varepsilon}^e = \sum_{I=1}^{4} \mathbf{B}_I(.) \cdot \mathbf{u}_I \ , \tag{1.7}$$

where we order the derivatives of the shape functions N_I of node number I in

$$\mathbf{B}_I = \begin{bmatrix} \frac{\partial N_I}{\partial x} & 0 \\ 0 & \frac{\partial N_I}{\partial y} \\ \frac{\partial N_I}{\partial y} & \frac{\partial N_I}{\partial x} \end{bmatrix} \ . \tag{1.8}$$

At the same time, we collect the four $\mathbf{B}_I (I = 1, 2, 3, 4)$ as

$$\mathbf{B}^e = [\mathbf{B}_1 \ \mathbf{B}_2 \ \mathbf{B}_3 \ \mathbf{B}_4] \tag{1.9}$$

in the block matrix $\mathbf{B}^e$. Reformulating (1.4) with these results (1.5)–(1.9), we obtain

$$\mathsf{A}\ \delta\mathbf{u}^{e\,\mathrm{T}} \cdot \int_v \mathbf{B}^{e\,\mathrm{T}} \cdot \check{\mathbb{C}} \cdot \mathbf{B}^e \cdot \mathbf{u}^e \,\mathrm{d}v - \delta\mathbf{u}^{e\,\mathrm{T}} \cdot \int_a \mathbf{t}\,\mathrm{d}a = 0 \tag{1.10}$$

with the assumption of discretising the variations $\delta\mathbf{u}^e$ and $\delta\check{\boldsymbol{\varepsilon}}^e$ in the same way as for $\mathbf{u}^e$ and $\check{\boldsymbol{\varepsilon}}^e$ itself, see e.g. (1.6). Please note, that we define by the A–operator the assembly procedure, which is in some sense a main task of any FE code. One can understand this operation as the computational counterpart of the discretization procedure with respect to the given

element–node–connectivity. We describe the treatment of that operation by DAEdalon in Sec. 4.4 in more detail.

By straight forward manipulations, (1.10) can be reformulated into

$$\delta\mathbf{u}^{\mathrm{T}} \cdot [\,\mathbf{K} \cdot \mathbf{u} - \mathbf{l}\,] = 0 \;, \tag{1.11}$$

where the stiffness matrix of the system $\mathbf{K}$ is identified as

$$\mathbf{K} = \mathop{\mathsf{A}}_{v} \int \mathbf{B}^{e\,\mathrm{T}} \cdot \check{\mathbb{C}} \cdot \mathbf{B}^{e}\, \mathrm{d}v \tag{1.12}$$

and the load vector $\mathbf{l}$ as

$$\mathbf{l} = \mathop{\mathsf{A}}_{a} \int \mathbf{t}\, \mathrm{d}a \;, \tag{1.13}$$

respectively. Please note the assembly of the global solution vector $\mathbf{u}$ in

$$\mathbf{u} = \mathop{\mathsf{A}} \mathbf{u}^{e} \tag{1.14}$$

and its virtual counterpart $\delta\mathbf{u}$, analogously. Assuming the vector $\delta\mathbf{u}$ to obtain any value, the product in (1.11) vanishes by taking

$$\mathbf{K} \cdot \mathbf{u} - \mathbf{l} = \mathbf{0} \;. \tag{1.15}$$

This enables us to solve (1.15) for $\mathbf{u}$ by

$$\mathbf{u} = \mathbf{K}^{-1} \cdot \mathbf{l} \tag{1.16}$$

for that linear case. Further possible treatments in the solution will be discussed in Sec. 2.7.

So, we obtain the global solution of the system given in the nodal displacements resulting in the deformed structure.

1.5 Run DAEdalon

We apply the described procedure to solve the stated problem of Sec. 1.2 by the FE method using DAEdalon. Doing so, one has generally to define the given problem in a prestructured way by different input files. This is often called *preprocessing*. DAEdalon expects at least information of the following form and structure by the given files in the subdirectory /input relative to our working directory. We propose the following input files for that example at www.DAEdalon.org/Examples.

Here, the mesh consists of 131 nodes and 110 elements, where here the first and the last four lines of the whole table of both files are given exemplary. The physical positions of the nodes are given line by line, where the three columns declare the x, y and z coordinate, respectively,

```
0.0000   0.0000   0.0
8.0000   0.0000   0.0
1.3333   0.0000   0.0
2.6667   0.0000   0.0
..   ..   ..
3.9103   1.1350   0.0
5.0820   2.2103   0.0
2.9001   0.8170   0.0
4.7273   1.3340   0.0
```

x y z–position, in mm

Fig. 1.3 Structure of Input File `node.inp`: Coordinates of Nodes #1 — # 131

in `node.inp`, see Fig. 1.3. In contrast, the declaration of the elements follow the rule `material number, local node 1, local node 2, local node 3, local node 4` for that typical 4–noded–element, where the local counting is in counterclockwise order. This is given in `el.inp`, which is illustrated in details in Fig. 1.4. Additionally, we declare linear elastic material behavior and the integration order within the described element by `mat1.inp` shown in Fig. 1.5, where especially the YOUNG modulus and the POISSON number are given. The basic geometrical informations describing the element behavior are given as `geom.inp` in Fig. 1.6. Nevertheless, the information about number of nodes and elements is possibly redundant with the information in `node.inp` and `el.inp`, respectively, so that we can set `0` at these positions in that file.

```
1    1    3    85   18
1    20   25   43   44
1    38   20   44   67
1    90   89   65   66
..   ..
1    125  121  110  124
1    111  125  124  12
1    120  125  111  126
1    120  118  121  125
```

material node 1 node 2 node 4 of element

Fig. 1.4 Structure of Input File `el.inp`: Material and Connectivity of Elements

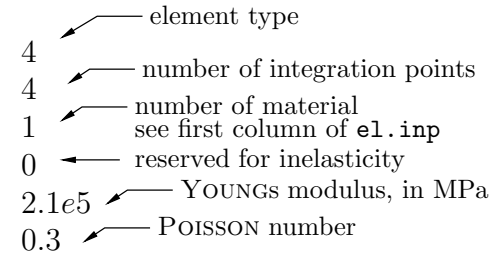

Fig. 1.5 Structure of Input File `mat1.inp`

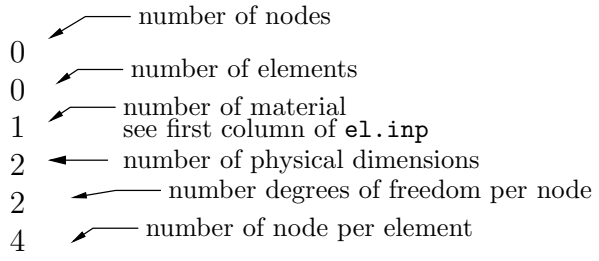

Fig. 1.6 Structure of Input File `geom.inp`

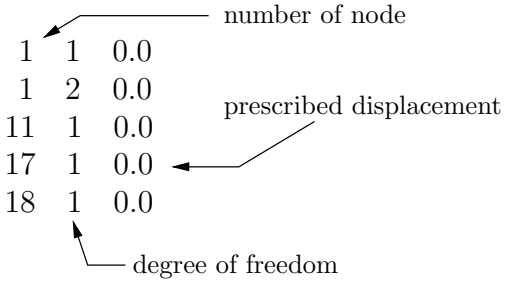

Fig. 1.7 Structure of Input File `displ.inp`

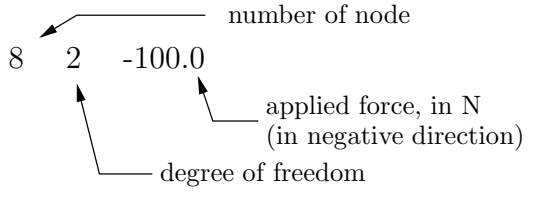

Fig. 1.8 Structure of Input File `force.inp`

At last, for this example, the boundary conditions of the problem have to be described by given displacements in `displ.inp`, see Fig. 1.7. as and by prescribed loads in `force.inp` as can be seen in Fig. 1.8. Now, we are able to start `DAEdalon` within a given MATLAB environment by prompting `dae`, which initialize the `DAEdalon`–FEM–system.

The instruction `lprob` loads the above given data defining our actual problem and `time` increments the time counter to 1.0.

A first check of our model can be done by visualizing the defined mesh, boundary conditions and applied forces by `mesh0`, `boun` and `forc`, respectively.

`go`, which consists of `stiffness, syst, solv, residuum`. The global assembly process of $\mathbf{K}$ and $\mathbf{r}$ is carried out in `syst`, where also the boundary conditions are respected.

With `solv` the global system as given in (1.15) is solved for $\mathbf{u}$.

One can check the "quality" of the solution by computing the global residuum

$$\mathbf{r} = \mathbf{K} \cdot \mathbf{u} - \mathbf{l} \,, \tag{1.17}$$

by `residuum`, which gives the scalar value $\mathbf{r}^{\mathrm{T}} \cdot \mathbf{r}$. A very small value for $\mathbf{r}^{\mathrm{T}} \cdot \mathbf{r}$ (computationally zero, e.g. 10^{-9} or smaller) indicates the fulfillment of condition (1.15).

The deformed structure can be visualized by `meshx` to get an impression of the solution. A scaling of the displacements is applied for the plot by setting the variable `defo_scal` to the scaling factor.

Visualization of the Results for Example 1.1

By default, we reserved `cont(1···2···3)` for the strains $\varepsilon_x, \varepsilon_y$ and $2\,\varepsilon_{xy}$, respectively, see (1.7), while in `cont(4···5···6)` the stress $\sigma_x, \sigma_y, \sigma_{xy}$ is stored.

In that sense, the σ_x stress distribution as shown in Fig. 1.9 can by obtained by `cont(4)` as contour plot.

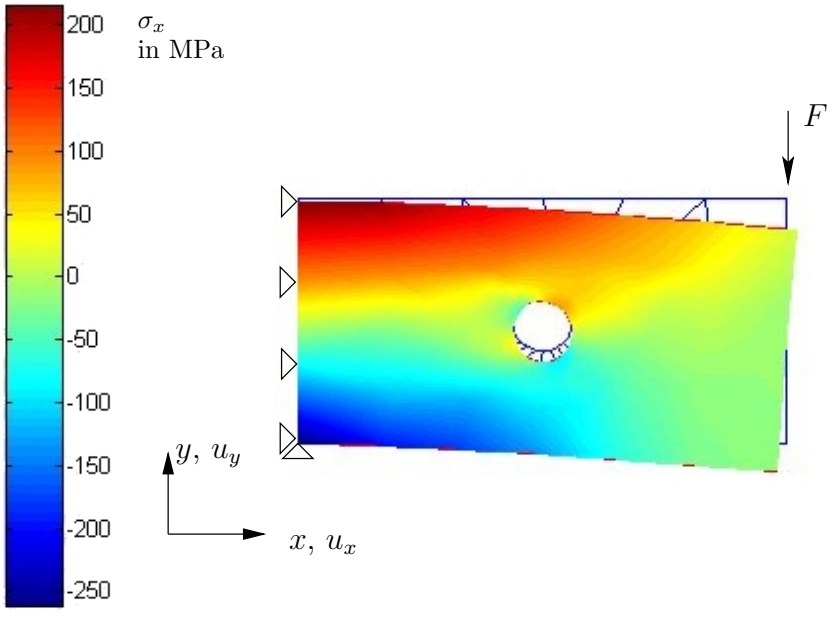

Fig. 1.9 Stress Distribution σ_x as Contour Plot in Deformed Structure. Deformation Scaled with Factor 30 by `defo_scal=30.0`. Undeformed Shape in Background.

2

Fundamentals of Solid (Continuum) Mechanics

We quickly resume the mathematical and mechanical preliminaries of modern textbooks like HOLZAPFEL [2000] in order to describe positions and motions of material points within any body of consideration. Further on, this enables us to define as well strains and stresses as gives finally a representation of mechanical equilibrium.

2.1 Vector–, Matrix– and Tensor–Notation

In treating the whole analysis in the orthogonal, threedimensional EUKLIDian space $\mathbb{E}^3$, we represent a vector $\mathbf{x}$ as

$$\mathbf{x} = x_1\mathbf{e}_1 + x_2\mathbf{e}_2 + x_3\mathbf{e}_3 = \sum_{i=1}^{3} x_i\mathbf{e}_i =: x_i\mathbf{e}_i = \{x_i\}\,, \tag{2.1}$$

where the $\mathbf{e}_1$, $\mathbf{e}_2$, $\mathbf{e}_3$ define the units vectors in $\mathbb{E}^3$. Additionally, we are able to apply the EINSTEIN convention in summing over double Roman indices.

In that sense, the *inner product* defines a scalar quantity by

$$c = \mathbf{a}^{\mathrm{T}} \cdot \mathbf{b} = \{a_i\, b_i\} \tag{2.2}$$

written on conformity with the MATLAB conventions, in spite of some authors just use $c = \mathbf{a} \cdot \mathbf{b}$ without the transpose operation $(\cdot)^{\mathrm{T}}$, which exchanges the rows and columns of an array.

Further on, we symbolize the *dyadic product* by

$$\mathbf{T} = \{T_{ij}\} = \mathbf{a} \otimes \mathbf{b} = \{a_i\, b_j\} \tag{2.3}$$

resulting in a second order tensor $\mathbf{T}$. The *trace* operation on a second order tensor quantity or just its $[3 \times 3]$ array representation is defined by

$$\begin{aligned} \mathrm{trace}(\mathbf{T}) = \mathrm{tr}(\mathbf{T}) &= T_{11} + T_{22} + T_{33} = \{T_{ii}\} \\ &= \{T_{ij}\,\delta_{ij}\} = \mathbf{T} : \mathbf{I}\,, \end{aligned} \tag{2.4}$$

which can also be seen as a double contraction of that tensor by the identity $\mathbf{I}$. For the index notation, here we introduce the KRONECKER symbol δ_{ij} for $\mathbf{I}$, which represents 1 if $i = j$ and 0 otherwise.

Exemplary, for such a 2nd order tensorial quantity we have a look onto a transformed representation. By application of an eigenvalue decomposition on $\mathbf{T}$, one may result in

$$\mathbf{T}^* = \mathbf{Q} \cdot \mathbf{T} \cdot \mathbf{Q}^{\mathrm{T}} \tag{2.5}$$

with $\mathbf{Q}^{\mathrm{T}} = \mathbf{Q}^{-1}$, $\det \mathbf{Q} = 1$ and symmetric $\mathbf{T} = \mathbf{T}^{\mathrm{T}}$, where $\mathbf{T}^*$ is a diagonalized array. The coefficients T_1^*, T_2^*, T_3^* of $\mathbf{T}^*$ are the eigenvalues of $\mathbf{T}$, whereas $\mathbf{Q}$ is composed by the eigenvectors $\mathbf{n}_1, \mathbf{n}_2$ and $\mathbf{n}_3$ in the sense $\mathbf{Q} = [\mathbf{n}_1 \mathbf{n}_2 \mathbf{n}_3]$.

Equivalently, we give the spectral decomposition

$$\mathbf{T} = \sum_{k=1}^{3} T_k^* \, \mathbf{n}_k \otimes \mathbf{n}_k \tag{2.6}$$

of the tensor $\mathbf{T}$, which is of special interest in Sec. 3.2.3.

Please remind that we neglect further on the explicit notation of the vector and tensor basis vectors $\mathbf{e}_1, \mathbf{e}_2$ and $\mathbf{e}_3$, which is justified in our sense by a consequent treatment in $\mathbb{E}^3$, see (2.1).

2.2 Kinematics and Deformation Gradient F

2.2.1 Definitions

The *actual* or *spatial* position of a material point may be given by the vector $\mathbf{x}(\mathbf{X}, t)$ and the corresponding displacement vector $\mathbf{u}(\mathbf{X}, t)$

$$\mathbf{x} = \mathbf{x}(\mathbf{X}, t) = \mathbf{X} + \mathbf{u}(\mathbf{X}, t) \tag{2.7}$$

with $\mathbf{X}$ as position vector in the undeformed reference configuration and t as time parameter, see Fig. 2.1.

For the sake of completeness and its relevance in the formulation of inelastic material behavior, we give here the spatial velocity field

$$\boldsymbol{v} = \dot{\mathbf{x}} = \frac{\partial \mathbf{x}}{\partial t} \tag{2.8}$$

as material time derivative of the spatial position $\mathbf{x}$.

Further on, the spatial gradient

$$\operatorname{grad} \boldsymbol{v} = \frac{\partial \boldsymbol{v}}{\partial \mathbf{x}} =: \mathbf{l} \tag{2.9}$$

of that field plays a crucial role in Sec. 3.2.3.

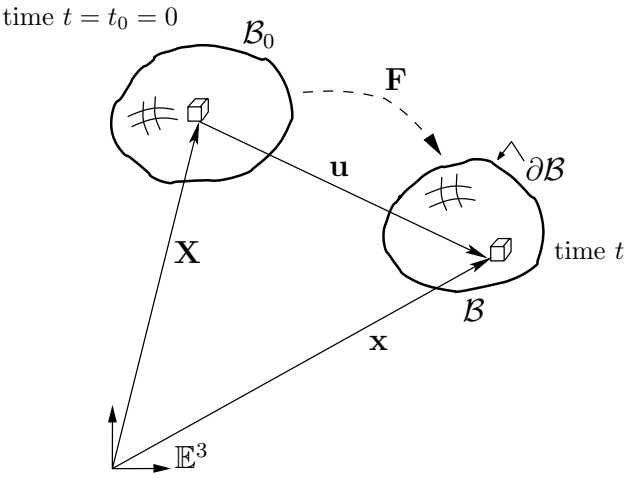

Fig. 2.1 Reference ($\mathcal{B}_0$) and Actual ($\mathcal{B}$) Configuration; their Mapping by **F** and resulting Displacement **u** between the Positions **X** and **x** in Physical (EUKLIDian) Space $\mathbb{E}^3$. The surface of the body is denoted by the $\partial\mathcal{B}$ symbol.

Finally, the deformation gradient is given by

$$\mathbf{F}(\mathbf{X}, t) = \mathrm{Grad}\,\mathbf{x}(\mathbf{X}, t) = \mathbf{I} + \mathrm{Grad}\,\mathbf{u}(\mathbf{X}, t)\,, \tag{2.10}$$

where the Grad () operator is defined as $\mathrm{Grad}() = \frac{\partial()}{\partial \mathbf{X}}$ with respect to the coordinates **X**. As mentioned above, for further derivations, we omit the time parameter t for simplicity in this treatise.

Nevertheless, here we do not concentrate on time dependent material behavior (*viscosity* or *rate–dependency*) and do not consider any dynamic behavior, although `DAEdalon` is proposed for such analysis, see *online manual* at `www.DAEdalon.org`.

Consequently, in the following we give quantities in the reference configuration $\mathcal{B}_0$ in capital letters, while such of the actual configuration $\mathcal{B}$ are given as lower case characters.

2.2.2 Properties and Derivatives of **F**

The deformation gradient **F** in (2.10) is the fundamental kinematic quantity describing the deformation of a material point and is – because of that – of special interest as well for the spatial description of each material point of a structure as for the representation of material laws.

Here, we give the well known polar decomposition

$$\mathbf{F} = \mathbf{R} \cdot \mathbf{U} = \mathbf{v} \cdot \mathbf{R}\,, \tag{2.11}$$

where $\mathbf{U}$ and $\mathbf{v}$ are known as *right stretch tensor* and *left stretch tensor*, respectively, and $\mathbf{R}$ denotes a rotational tensor with $\det \mathbf{R} = 1$ and $\mathbf{R}^{\mathrm{T}} \cdot \mathbf{R} = \mathbf{I}$.

Exemplary, here we give a very efficient realization of a computation of the polar decomposition based on strain tensors in (2.17) resulting $\mathbf{U}$, $\mathbf{R}$ and $\lambda_1, \lambda_2, \lambda_3$ known as principal stretches, see Sec. 2.3.2.

```
function [U,R,lambda] = polardecompF(F)

% right CAUCHY-GREEN tensor
C=F'*F;

% eigenvalue decomposition
[N,lambda] = eig(C);

% principal stretches
lambda = sqrt( diag(lambda) );

% right stretch tensor
U = lambda(1) * N(:,1)*N(:,1)' ...
  + lambda(2) * N(:,2)*N(:,2)' ...
  + lambda(3) * N(:,3)*N(:,3)';

% rotation tensor
R = F*inv(U);
```

As we see later on, the determinant

$$J := \det \mathbf{F} = \frac{\mathrm{d}v}{\mathrm{d}V} \tag{2.12}$$

plays a crucial role in the formulation of material laws, as it also represents the ratio of the actual to the reference volume under deformation, see Fig. 2.1. That quantity if often associated with the name of Jacobi. [1] In this treatise we do not concentrate on time dependent material behavior, see for more details on that topic, and especially on the treatment of time–independent finite plasticity the literature following Weber & Anand [1990] and Simo [1992].

The material time derivative of $\mathbf{F}$ is given by

$$\dot{\mathbf{F}} = \operatorname{Grad} \boldsymbol{v} = \mathbf{l} \cdot \mathbf{F}\,, \tag{2.13}$$

where $\mathbf{l}$ is given by (2.9). Again, we mention the importance of (2.9) and the property (2.13) with respect to modern formulations of inelastic material behavior, see Sec. 3.2.3. Especially, the symmetric part of $\mathbf{l}$

[1] Please do not mix up the tensor $\mathbf{v}$ in (2.11) with the velocity $\boldsymbol{v}$ some lines above or our symbol v for the actual volume under consideration.

$$\mathbf{d} := \frac{1}{2}(\mathbf{l} + \mathbf{l}^{\mathrm{T}}) \tag{2.14}$$

represents the spatial deformation velocity and is often used as strain rate tensor.

2.2.3 Implementation

In anticipation of concrete element formulations in Sec. 2.10, we want demonstrate exemplary the computation of the crucial deformation tensor within a typical element formulation consisting of *nel* nodes with respective shape functions $N_I, I = 1, ..., nel$.

For a discrete representation of (2.10), we rewrite the Grad operation and end up with the matrix

$$\mathbf{F} = \mathbf{I} + \sum_{I=1}^{nel} \frac{\partial N_I}{\partial \mathbf{X}} \cdot \mathbf{u}_I \,, \tag{2.15}$$

which is given in `DAEdalon` in just one line as

```
F = eye(3) + u_elem' * dshape; ,
```

whenever the derivatives of N_I are given ordered in `dshape`.

This example shows very clear the idea and structure of MATLAB based computation. Please review for that task the complete `DAEdalon` script `defgrad_3d.m`.

Analogously, we are able to compute $\mathbf{F}$ in terms given in the actual configuration $\mathcal{B}$ by

$$\mathbf{F} = \left[\mathbf{I} - \sum_{I=1}^{nel} \frac{\partial N_I}{\partial \mathbf{x}} \cdot \mathbf{u}_I \right]^{-1} \tag{2.16}$$

as e.g. given in `defgrad_x.m`.

2.3 Strain– and Stress–Tensors

2.3.1 A Selection

We restrict ourselves to two representing strain tensors, named GREEN–LAGRANGE tensor $\mathbf{E}$ and ALMANSI tensor $\mathbf{e}$, acting on the reference configuration $\mathcal{B}_0$ and the current configuration $\mathcal{B}$, respectively, see Fig. 2.1.

These are defined by

$$\mathbf{E} := \frac{1}{2}(\mathbf{C} - \mathbf{I}) \tag{2.17}$$

with the *right* CAUCHY–GREEN *tensor* $\mathbf{C} := \mathbf{F}^{\mathrm{T}} \cdot \mathbf{F}$ and

$$\mathbf{e} := \frac{1}{2}(\mathbf{I} - \mathbf{b}^{-1}) \tag{2.18}$$

with the *left* CAUCHY–GREEN *tensor* $\mathbf{b} := \mathbf{F} \cdot \mathbf{F}^{\mathrm{T}}$ as symmetric tensors out of the deformation gradient $\mathbf{F}$ in (2.10).

2.3.2 Invariants and Derivatives of Strain Tensors

These two tensors $\mathbf{C}$ and $\mathbf{b}$ are of special interest in respect of the formulation of material models in Chap. 3. Therefore, we view on their invariants and the derivatives of these invariants with respect to the two tensors itself. Further on, we give the invariants also as functions of the three principal stretches $\lambda_1, \lambda_2, \lambda_3$, whose squares $\lambda_1^2, \lambda_2^2, \lambda_3^2$ are the eigenvalues of $\mathbf{C}$ and $\mathbf{b}$, respectively, see Sec. 2.1 and especially (2.6).

In detail, the first invariant is given by

$$I_1(\mathbf{C}) = I_1(\mathbf{b}) = \operatorname{trace}\mathbf{C} = \operatorname{trace}\mathbf{b} = \lambda_1^2 + \lambda_2^2 + \lambda_3^2 \ , \tag{2.19}$$

while the second and third invariant write as

$$I_2(\mathbf{C}) = I_2(\mathbf{b}) = \frac{1}{2}\left[(\operatorname{trace}\mathbf{C})^2 + \operatorname{trace}\mathbf{C}^2\right] \tag{2.20}$$

$$= \operatorname{trace}\mathbf{C}^{-1} \det\mathbf{C} = \lambda_1^2\,\lambda_2^2 + \lambda_1^2\,\lambda_3^2 + \lambda_1^2\,\lambda_3^2 \tag{2.21}$$

and

$$I_3(\mathbf{C}) = I_3(\mathbf{b}) = \det\mathbf{C} = J^2 = \lambda_1^2\,\lambda_2^2\,\lambda_3^2 \ . \tag{2.22}$$

In that sense, one obtains the derivatives of (2.19)–(2.22) by

$$\frac{\partial I_1}{\partial \mathbf{C}} = \frac{\partial \operatorname{trace}\mathbf{C}}{\partial \mathbf{C}} = \frac{\partial \mathbf{I} : \mathbf{C}}{\partial \mathbf{C}} = \mathbf{I} \ , \tag{2.23}$$

$$\frac{\partial I_2}{\partial \mathbf{C}} = \frac{1}{2}(2\operatorname{trace}\mathbf{C}\,\mathbf{I} - \frac{\partial \operatorname{trace}\mathbf{C}^2}{\partial \mathbf{C}}) = I_1\,\mathbf{I} - \mathbf{C} \tag{2.24}$$

and

$$\frac{\partial I_3}{\partial \mathbf{C}} = I_3\,\mathbf{C}^{-1} \ . \tag{2.25}$$

2.3.3 Stress Representation — VOIGT Notation

From the view of the (solid) continuum mechanics a body $\mathcal{B}$ may be loaded by external forces $\mathbf{b}$ acting on volume particles and by surface loads $\mathbf{t}$. Both cause internal forces $\mathbf{F}_{int}$ in that body by interaction of neighboring parts of the body volume. In that sense, we speak of *area based forces* or *stresses*. So, we define the stress vector

$$\mathbf{t} = \lim_{\Delta a \to 0} \frac{\mathbf{F}_{int}}{\Delta a} \tag{2.26}$$

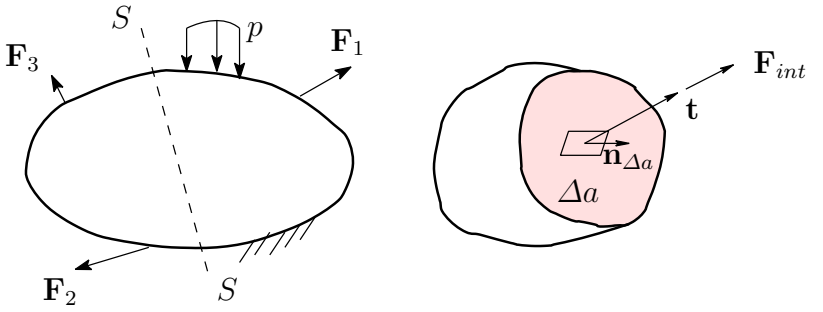

Fig. 2.2 Arbitrary Cut $S - S$ through Loaded Body $\mathcal{B}$

in the limit of the considered area partition Δa. This situation is illustrated in Fig. 2.2, where $\mathbf{n}_{\Delta a}$ describes the outward normal to the area segment of interest.

At the same time, following the so–called CAUCHY *theorem*

$$\mathbf{t} = \boldsymbol{\sigma} \cdot \mathbf{n}_{\Delta a} \ , \tag{2.27}$$

the stress vector $\mathbf{t}$ is also represented by the second order *stress tensor* $\boldsymbol{\sigma}$ in any material point of the body under a cutting plane defined by $\mathbf{n}_{\Delta a}$.

In the scope of that booklet, we restrict further on to a symmetric representation of the CAUCHY stress tensor $\boldsymbol{\sigma}$, so that we deal with six components of the originally $3^2 = 9$ coefficients of that second order tensor in three spatial dimensions.

Following the so–called VOIGT notation, we represent the stress tensor by its coefficients in column vector order as

$$\check{\boldsymbol{\sigma}} = \begin{bmatrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{33} \\ \sigma_{12} \\ \sigma_{23} \\ \sigma_{13} \end{bmatrix} \ . \tag{2.28}$$

Remark

We order the six coefficients of any symmetric strain measure $\boldsymbol{\varepsilon}$ as linearized form of $\mathbf{E}$ and $\mathbf{e}$, see Sec. 2.3.1, in the same scheme as shown above, like

$$\check{\boldsymbol{\varepsilon}} = \begin{bmatrix} \varepsilon_{11} \\ \varepsilon_{22} \\ \varepsilon_{33} \\ 2\,\varepsilon_{12} \\ 2\,\varepsilon_{23} \\ 2\,\varepsilon_{13} \end{bmatrix} \ . \tag{2.29}$$

For that reason, the ordering scheme of the coefficients of a *forth order material tensor* $\mathbb{D}$, see Chap. 3, follows strictly through that motivation and storing scheme.

2.4 Rate of Strain Energy — Stress Power, Internal Energy Turn Over

At last, we give exemplary representations for the internal energy turn over. This expression describes the internal energy turn over physically and documents the conjugated stress and strain tensors used in the later formalism in Sec. 2.5 and especially in Chap. 3. Following the derivations in e.g. Holzapfel [2000] on the principle of mechanical energy, which we omit here again for simplicity, the (internal) stress power per volume is given by

$$\pi_{\text{int}} = \frac{1}{v}\,\mathcal{P}_{\text{int}} = \boldsymbol{\sigma} : \mathbf{d} \tag{2.30}$$

in terms of the Cauchy stress in (2.27) and the strain rate (2.14). Applying some manipulations recombining (2.30) by $\mathbf{F}$ and $\dot{\mathbf{F}}$, we obtain the equivalent expression with respect to the reference volume as

$$\pi^0_{\text{int}} = \frac{1}{V}\,\mathcal{P}_{\text{int}} = \mathbf{S} : \dot{\mathbf{E}} \tag{2.31}$$

with the rate of (2.17). Here, we are able to define the symmetric *second* Piola–Kirchhoff stress tensor by

$$\mathbf{S} := J\,\mathbf{F}^{-1} \cdot \boldsymbol{\sigma} \cdot \mathbf{F}^{-\mathrm{T}} \tag{2.32}$$

as power conjugate quantity to $\dot{\mathbf{E}}$. At that point, we again see the advantage of the Voigt storing scheme following (2.28) and (2.29), because in that sense, the double contraction in (2.30) ending in a scalar valued function like π_{int} states as a simple vector multiplication.

2.5 Variational Principle and Weak Form

Starting point of our finite element discretization is the weak form of equilibrium, given here in a spatial description as

$$g(\mathbf{u}, \delta\mathbf{u}) := \int_{\mathcal{B}} \boldsymbol{\sigma} : \operatorname{grad}\delta\mathbf{u}\,\mathrm{d}v - \int_{\partial\mathcal{B}_\sigma} \mathbf{t}_L \cdot \delta\mathbf{u}\,\mathrm{d}a = 0\,, \tag{2.33}$$

where $\mathbf{u} = \mathbf{x} - \mathbf{X}$ denotes the displacement vector of a material point represented by $\mathbf{X}$ in the reference configuration toward a position $\mathbf{x}$ of the same point in the current configuration, $\delta\mathbf{u}$ is the first variation of the displacement field. With $\boldsymbol{\sigma}$ the Cauchy stress tensor is characterized and $\mathbf{t}_L = \boldsymbol{\sigma} \cdot \mathbf{n}$

are the prescribed tractions acting on the loaded boundary $\partial\mathcal{B}_\sigma$ of the body with (outer) normal vector $\mathbf{n}$ in the current configuration $\mathcal{B}$. Equivalently, (2.33) can be obtained by the well–known derivation from the strong form of equilibrium (*local balance of momentum*).

Remark: There is no need to found a finite element formulation on a variational principle. Some other applications are known in literature. On the other hand, there a many more types and enhacements on such 'simple' representations like (2.33); but for our basical explanations and discription it seems to be much more attractive to come from that ...

2.6 Discretization in Space

2.6.1 Preparation and Rearrangement of Equations

In order to prepare (2.33) for a computational application by matrix–vector representations, we reformulate and rearrange the expressions. Essentially, the second order stress tensor $\boldsymbol{\sigma}$ is given as outlined in (1.2) in vector form in a sense following the VOIGT notation $\check{\boldsymbol{\sigma}}$. In doing so, we are able to give

$$g(\mathbf{u}, \delta\mathbf{u}) = \int_{\mathcal{B}} \mathrm{grad}\delta\mathbf{u} \cdot \check{\boldsymbol{\sigma}} \; \mathrm{d}v - \int_{\partial\mathcal{B}_\sigma} \delta\mathbf{u} \cdot \mathbf{t}_L \, \mathrm{d}a = 0 \,, \tag{2.34}$$

in pure scalar products. For simplicity, the further treatment is given on the so–called element level, from which the mechanical problem of (2.33) and (2.34) respectively, is given by the assembly procedure denoted by

$$g(\mathbf{u}, \delta\mathbf{u}) \approx \mathsf{A} \; g^{elmt}(\mathbf{u}, \delta\mathbf{u}) \,. \tag{2.35}$$

Therefore, we refer the Sec. 4.4.1, where the assembly procedure of a finite element system is described. At this point of description, we emphasize again the advantages of using MATLAB as basis for `DAEdalon`, where matrix–vector operations and especially matrix reorganisation techniques are core operations of that system. From this point of view, it is easy to accept such a system as best tool for learning the basics of the finite element method. So, from now on, we concentrate on *element* expressions like $g^{elmt}(\cdot)$ and have the A–operator in mind. With the exception of special applications (see e. g. nonlocal material behaviour, BAASER & TVERGAARD [2003]), that focus is sufficient in FE–applications. Accepting that, we give

$$g^{elmt}(\mathbf{u}, \delta\mathbf{u}) = \int_{\mathcal{B}} \mathrm{grad}\delta\mathbf{u}^e \cdot \check{\boldsymbol{\sigma}}^e \; \mathrm{d}v^{elmt} \tag{2.36}$$

equivalently to the above formulations as contribution of each element to the global system, where we explicitly exclude the second term in (2.34), which is given also on the global level to the overall system, see Sec. 4.4.1.

2.6.2 Linear Shape Functions

We have to represent the body of consideration $\mathcal{B}$ by discrete objects which can be handled by a computer system. Beside any mathematical formalism, a vivid counterpart is the cutting of the body into (conforming) subdomains — the finite elements. These elements are defined by *nodes*, at the corners, at the element edges or partly within the element domain and so–called *shape functions* based on the nodal positions.

Despite there is a wide literature focused on that topic and the accuracy of such approximations, see e.g. DHATT & TOUZOT [1985], here we concentrate on linear functions N_I, which enable us to discretize the functions $\mathbf{u}$ and $\partial\mathbf{u}$, respectively. Besides that, we avoid here any further examination on the construction of that functions, which is discussed intensively during the history and development of the FE method. We refer to the famous text books such as DHATT & TOUZOT [1985] and HUGHES [2000], where different approaches such as *serentipity*– and LAGRANGE–types are contrasted and discussed. In our case, just using linear shape functions, these types results in the same functions, which can be regarded as constructed of prototype terms of the form $\frac{1}{2}(1 \pm \xi)$.

Clearly, the shape functions have to fulfill the two conditions

- *The value at the relating node has to be 1, at the other* $nel - 1$ *nodes of the element the value of the shape function has to be 0.*
- *The sum of all nel shape functions at any position* ξ *in an element has to be 1:* $\sum_{I=1}^{nel} N_I(\boldsymbol{\xi}) = 1$.

that we can use them for our approximation.

To demonstrate that, we firstly consider the two shape functions of the later described 2–noded truss element of length l_e:

$$N_1(\xi) = 1 - \frac{\xi}{l_e} \quad \text{and} \quad N_2(\xi) = \frac{\xi}{l_e}\,, \tag{2.37}$$

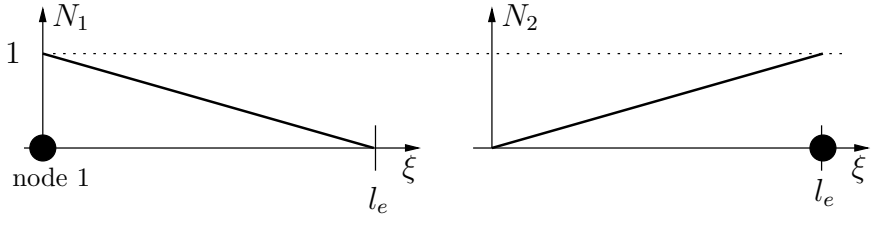

Fig. 2.3 Linear Shape Functions for 2–Noded Element of Length l_e

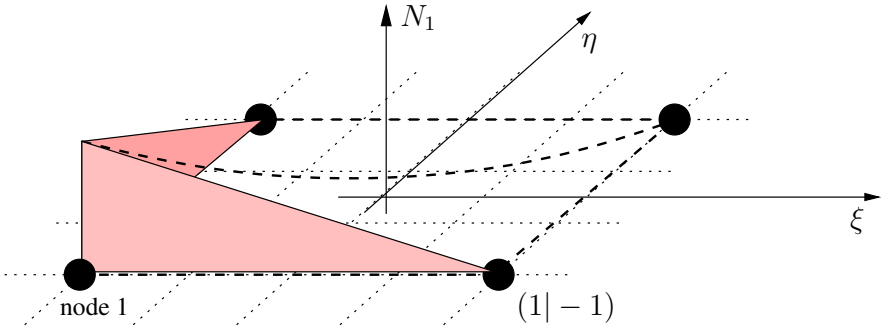

Fig. 2.4 Linear Shape Function N_1 for 4–Noded Element in Reference Space $[\xi, \eta]$

which fulfill the above mentioned conditions in the simplest way and are refered in Sec. 4.6.4.

Equivalently, we built up the shape functions of the 4–noded (protype) element by

$$N_1(\xi,\eta) = \frac{1}{2}(1-\xi) \cdot \frac{1}{2}(1-\eta) = \frac{1}{4}(1-\xi)(1-\eta) \tag{2.38}$$

for the reference element with nodes at $(\pm 1|\pm 1)$ as given in Fig. 2.4. By that and according to the isoparametric concept, the field of displacements over the domain $\mathcal{B}$ is described by

$$\mathbf{u}^e = \sum_{I=1}^{nel} N_I(\boldsymbol{\xi})\, \mathbf{u}_I \ , \tag{2.39}$$

where I counts over all nel element nodes and $\mathbf{u}_I$ represents the discrete nodal displacement vector.

2.6.3 Derivatives of the Shape Functions

The depiction of the derivatives $\frac{\partial N_I}{\partial \mathbf{x}}$, needed e.g. in (2.15) or later in Sec. **??**, is given by

$$\frac{\partial N_I}{\partial \mathbf{x}} = \mathcal{J}^{-\mathrm{T}} \frac{\partial N_I}{\partial \boldsymbol{\xi}} \ , \tag{2.40}$$

where we have to consider a mapping by the JACOBIan $\mathcal{J}$ of that transformation because the functions N_I are defined in their reference space $\boldsymbol{\xi}$. So, we can compute the partial derivatives quite easily, resulting examplary for (2.4) in

$$\frac{\partial N_1(\xi,\eta)}{\partial \xi} = -\frac{1}{4}(1-\eta) \ . \tag{2.41}$$

Applying the chain rule

$$\begin{aligned}\frac{\partial N_I}{\partial \xi} &= \frac{\partial N_I}{\partial x}\frac{\partial x}{\partial \xi} + \frac{\partial N_I}{\partial y}\frac{\partial y}{\partial \xi} \\ \frac{\partial N_I}{\partial \eta} &= \frac{\partial N_I}{\partial x}\frac{\partial x}{\partial \eta} + \frac{\partial N_I}{\partial y}\frac{\partial y}{\partial \eta},\end{aligned} \tag{2.42}$$

we obtain $\mathcal{J}$ as

$$\mathcal{J} = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial x}{\partial \eta} \\ \frac{\partial y}{\partial \xi} & \frac{\partial y}{\partial \eta} \end{bmatrix}. \tag{2.43}$$

Here, the coefficients of $\mathcal{J}$ are computed by e.g.

$$\frac{\partial x}{\partial \xi} = \sum_{I=1}^{nel} \frac{\partial N_I(\xi,\eta)}{\partial \xi}\, x_I \tag{2.44}$$

following the isoparametric concept, which enables us in the end to obtain the derivatives of the shape functions with respect to the physical coordinates $\mathbf{x}$ in (2.40).

Remark
As J in (2.12) gives the volume ratio $\frac{\mathrm{d}v}{\mathrm{d}V}$, here $\mathcal{J}$ scales the physical element volume (or area) to the reference element volume (or area), which has in the here considered case of a 4–noded 2d element the value of 4. That fact is an excellent possibility to check the own element coding with respect to the mentioned volume (or area) transformation.

Basically, in `DAEdalon` we provide the functions

`shape_quad / _tetr / _tri_lin / _quad`

for the different element types. Without any restriction, the user can add his own shape function routines answering the values N_I and $\mathbf{B}$ depending on the physical coordiantes $\mathbf{X}$ or $\mathbf{x}$ with respect to the reference or actual configuration, respectively.

We demonstrate the coding of the above shown functions $N_1 ... N_4$, see Fig. 2.4, in `shape_quad_lin.m`, which also computes the derivatives $\frac{\partial N_I}{\partial \mathbf{x}}$. The structure of that function follows the descripton above:

```
function [shape,dshape,j] = shape_quad_lin(x,coor)

xsi=coor(1);
eta=coor(2);
shape(1) = 0.25*(1.0-xsi)*(1.0-eta);
shape(2) = 0.25*(1.0+xsi)*(1.0-eta);
shape(3) = 0.25*(1.0+xsi)*(1.0+eta);
shape(4) = 0.25*(1.0-xsi)*(1.0+eta);
```

```
% derivatives dN/dxsi ...
nshape(1,1) = -0.25*(1.0-eta);
nshape(1,2) = -0.25*(1.0-xsi);
nshape(2,1) =  0.25*(1.0-eta);
nshape(2,2) = -0.25*(1.0+xsi);
nshape(3,1) =  0.25*(1.0+eta);
nshape(3,2) =  0.25*(1.0+xsi);
nshape(4,1) = -0.25*(1.0+eta);
nshape(4,2) =  0.25*(1.0-xsi);

% dx/dxsi = sum(N_xsi*x)
x_xsi=x'*nshape;
xsi_x = inv(x_xsi);

% transformation
dshape=nshape*xsi_x;
j= det(x_xsi);
```

Please note, that the order of the blocks in the function refer to (2.38), (2.41), (2.44) and (2.40), respectively.

2.6.4 Volume Integration

In the following of (2.33) up to (2.36), we have to compute volume integrals numerically. Due to its numerical accuracy, the GAUSS quadrature is the most famous rule and is widely achieved for such applications, because of its high accuracy as approximation rule for integrals. Without loss of generality we can focus on one–dimensional integrations in order to demonstrate the idea of such approximations. Additionally, we are able to transform any integral with arbitrary limits to a standardized form

$$I = \int_{-1}^{+1} g(\xi)\,\mathrm{d}\xi \tag{2.45}$$

by simple substitution of the integrant to ξ in the given limits from -1 to +1. Henceforth, we want to approximate I best possible by a summation scheme, that minimizes

$$I - \sum_{j=1}^{n} w_j\, g(\xi_j) \rightarrow min\ , \tag{2.46}$$

where we can choose n *weights* w_j and n *supporting points* ξ_j for the evaluation of $g(.)$.

Example: In the case of the cubic polynom $g(\xi) = k_0 + k_1\xi + k_2\xi^2 + k_3\xi^3$ with four coefficients $k_1 ... k_4$, we obtain four conditions for an *optimal* determination

of two weights w_1, w_2 and two supporting points ξ_1, ξ_2 in the case of $n = 2$ in (2.46):

$$\int_{-1}^{+1} k_0 + k_1\xi + k_2\xi^2 + k_3\xi^3 \,\mathrm{d}\xi \stackrel{!}{=} w_1\, g(\xi_1) + w_2\, g(\xi_2) \tag{2.47}$$

Comparing coefficients in (2.47) gives

$$\begin{aligned} 2k_0 &= (w_1 + w_2)\, k_0 \\ \frac{2}{3}k_2 &= (w_1\xi_1^2 + w_2\xi_2^2)\, k_2 \\ 0 &= (w_1\xi_1 + w_2\xi_2)\, k_1 \\ 0 &= (w_1\xi_1^3 + w_2\xi_2^3)\, k_3 \ , \end{aligned} \tag{2.48}$$

resulting in four equations $(2.48)_1$ to $(2.48)_4$ for the four unknowns w_1, w_2, ξ_1 and ξ_2. In order to result in a symmetric distribution of the both supporting points ξ_1 and ξ_2, see Fig. 2.5, with equal weighting factors w_1, w_2, so that

$$\begin{aligned} w &:= w_1 = w_2 \\ \xi &:= \xi_1 = -\xi_2 \ , \end{aligned} \tag{2.49}$$

we obtain directly

$$\begin{aligned} w &= w_1 = w_2 = 1 \\ \xi &= \xi_1 = -\xi_2 = \frac{1}{\sqrt{3}} \approx 0.57735 \end{aligned} \tag{2.50}$$

as solution of the system (2.48). So, we end up with a quite imposing result for that integration scheme by just two terms in (2.47), which is the most used in `DAEdalon` for linear elements. In the same manner, one can discuss

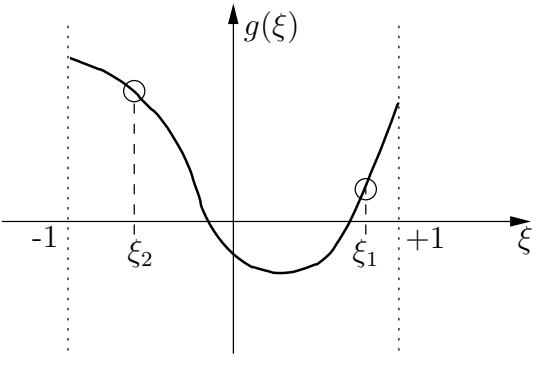

Fig. 2.5 Integration of an Arbitrary Function by a 2nd Order Gauss Scheme

higher order approximation schemes in the sense of chosing more supporting points in the interval $[-1;+1]$. Nevertheless, for our applications the demonstrated 2nd order scheme is adequate and is expanded in the same way for 2d– and 3d–integration.

Remark
The GAUSS integration scheme is known to be *exact* for polynoms of order $2n-1$, if n is the number of integration points in the sum in (2.46). Therefore it is the integration scheme of that type with the highest accuracy.

In `DAEdalon` we use functions of the type

```
function [gpcoor, gpweight]= gp_quad_lin
coor = 1.0/sqrt(3.0);
gpcoor = [ -coor -coor; ...
           +coor -coor; ...
           +coor +coor; ...
           -coor +coor];
gpweight=[1;1;1;1];
```

to obtain the supporting points and the weights on element level. Exemplary, we give again the version for our 4–noded prototype element with a 2×2 integration scheme in ξ– and η–direction, respectively.

Looking at the discretization of the leading FE equations such as (2.36), we have to substitute the integrals by the above mentioned summations, so that — in the view of (2.36) — the expression

$$g^{elmt}(\mathbf{u}, \delta\mathbf{u}) \approx \delta\mathbf{u}^e \cdot \sum_n \mathbf{B}_n^{e\,T} \cdot \check{\boldsymbol{\sigma}}_n^e \; w_n \tag{2.51}$$

results, where in the $\mathbf{B}$–matrix the derivatives of the shape functions N_I are ordered in such way, that the stress vector (in VOIGT notation, see (2.28)) is multiplied by a matrix–vector operation. Again, n counts for the integration points given by specifications to minimize the approximation error depending on the order of the integrating functions, see (2.46) above or DHATT & TOUZOT [1985] and HUGHES [2000].

2.7 General Treatment of Nonlinearities — Solution by NEWTON–Procedure

In order to describe and handle as well as linear and generally nonlinear problems in the same manner, we discuss the solution procedure in the form of an iteration scheme. That proceeding needs a linearization of the (algorithmic) setting of the given system (2.33) with respect to the global unknown represented by the vector $\mathbf{u}$ in (1.14).

Therefore, we approximate the functional (2.33) at a known equilibrium state $\hat{\mathbf{u}} = \mathbf{u} - \Delta\mathbf{u}$ by a TAYLOR series and break it off after the linear term, which ends in

$$g(\mathbf{u}, \delta\mathbf{u})|_{u=\hat{u}} \approx g(\hat{\mathbf{u}}, \delta\mathbf{u}) + \Delta g(\hat{\mathbf{u}}, \delta\mathbf{u}) \cdot \Delta\mathbf{u} = 0\,. \tag{2.52}$$

2.7.1 Linearization

Linearization at a known position $\mathbf{X} + \hat{\mathbf{u}}$ with respect to the current deformation state and rearrangement leads to the following representation of the element stiffness

$$\Delta g^{elmt}(\hat{\mathbf{u}}, \delta\mathbf{u}) = \int_{\mathcal{B}^{elmt}} (\Delta\boldsymbol{\sigma} + \underbrace{\mathrm{grad}\Delta\mathbf{u} \cdot \boldsymbol{\sigma}}_{\mathbf{k}_{geom}}) : \mathrm{grad}\,\delta\mathbf{u}\ \mathrm{d}v^{elmt}\,, \tag{2.53}$$

where $\Delta(\bullet)$ denotes the linearization operator and $\Delta\mathbf{u} = \mathbf{u} - \hat{\mathbf{u}}$ the increment of the displacement field $\mathbf{u}$. The right part of (2.53) results in the element stiffness *matrix* $\mathbf{K}^{elmt}$ for the discretized setting, where $\mathbf{K}^{elmt}$ obviously consists of two parts. The first part $\mathbf{K}_{mat}$ is obtained from the consistent linearization of the material model getting $\Delta\boldsymbol{\sigma}$, see Chap. 3, and the second part $\mathbf{K}_{geom} = \int \mathbf{k}_{geom} \mathrm{d}v^{elmt}$ comes solely from the linearization of the used strain measure at the computed stress state $\boldsymbol{\sigma}$.

2.7.2 Iteration

The further argumentation and application goes directly along with the numerical implementation of the solution technique, which is shown in Sec. 4.4.2. We rearrange (2.52) with the information (2.53) in the form

$$\delta\mathbf{u} \cdot \{\ \Delta g(\hat{\mathbf{u}}) \cdot \Delta\mathbf{u} + g(\hat{\mathbf{u}})\ \} = 0 \tag{2.54}$$

with $\delta\mathbf{u}$ anymore arbitrary, which allows to zero the expression within the curly bracket by solving

$$\Delta g(\hat{\mathbf{u}}) \cdot \Delta\mathbf{u} = -g(\hat{\mathbf{u}}) \tag{2.55}$$

for $\Delta\mathbf{u}$ in an iterative manner and by respecting the update $\mathbf{u} = \hat{\mathbf{u}} + \Delta\mathbf{u}$ as given above. Please remind that in the general case both $\Delta g(\hat{\mathbf{u}})$ and $g(\hat{\mathbf{u}})$ may be additionally functions of $\Delta\mathbf{u}$ as can be seen in (2.53) for the first case. Hence, the iteration loop solving (2.55) within the update of $\mathbf{u}$ has to be applied as long as the condition (2.52) is not achieved within a given tolerance.

3

Constitutive Behavior

3.1 The Materials Point of View

The description of any material behavior within a finite element simulation requires a clearly structured interface within an element formulation. As stated before in (2.36), in Sec. 2.6.4 and in (2.53), the (local) material behavior is respected and needed for at the integration points while the integration loop on element level. According to the presented modular structure of `DAEdalon`, we intend to define any constitutive model by the `function MATMOD`, where the deformation gradient $\mathbf{F}$ and the *history database* is given as input and the material response is given out in terms of the stress tensor and the material (tangent) modulus. In that sense, linear or nonlinear material behavior is treated in the same way. So, the user has all possibilities in defining any deformation measure as function of the deformation gradient $\mathbf{F}$ and the history and take care for the conjugated stress tensor and its derivative. Please note, that the overall convergence behavior of a finite element solution procedure depends dramatically on the *right* formulation of the stress response and its derivative in form of the material modulus.

3.1.1 Stress Response

Basically, one has the free choice in formulating the material behavior with respect to any configuration. There are many discussions on that topic, each with advances and each with negative aspects, but the overlaying element formulation has to be respected to take care on the energy turn over expressed by the tensors given by the material model, see Sec. 2.4.

3.1.2 The Material Modulus $\mathbb{D}$

The crucial task in the formulation and assembly of $\mathbf{K}_{mat}$ in (2.53) is the representation of the derivative $\mathbb{D}$ of the stress response with respect to the

applied deformation measure in the algorithmic setting. We illustrate here the results of that procedure for two typical material behaviors, namely hyperelasticity in its simplest form and finite plasticity, in the following.

Generally, we have a look on the structure and the implementation of typical representations of forth order tensors like

$\mathbf{I} \otimes \mathbf{I}$ as

$$\begin{bmatrix} 1&1&1&0&0&0 \\ 1&1&1&0&0&0 \\ 1&1&1&0&0&0 \\ 0&0&0&0&0&0 \\ 0&0&0&0&0&0 \\ 0&0&0&0&0&0 \end{bmatrix} \quad ; \tag{3.1}$$

$\mathbb{1}$ as

$$\begin{bmatrix} 1&0&0&0&0&0 \\ 0&1&0&0&0&0 \\ 0&0&1&0&0&0 \\ 0&0&0&1&0&0 \\ 0&0&0&0&1&0 \\ 0&0&0&0&0&1 \end{bmatrix} \tag{3.2}$$

and term like $\mathbf{A} \otimes \mathbf{A}$ for symmetric $\mathbf{A} = \{a_{ij}\}$ as

$$\begin{bmatrix} a_{11}a_{11} & a_{11}a_{22} & a_{11}a_{33} & 2a_{11}a_{12} & 2a_{11}a_{23} & 2a_{11}a_{13} \\ a_{22}a_{11} & a_{22}a_{22} & a_{22}a_{33} & 2a_{22}a_{12} & 2a_{22}a_{23} & 2a_{22}a_{13} \\ a_{33}a_{11} & a_{33}a_{22} & a_{33}a_{33} & 2a_{33}a_{12} & 2a_{33}a_{23} & 2a_{33}a_{13} \\ a_{12}a_{11} & a_{12}a_{22} & a_{12}a_{33} & 2a_{12}a_{12} & 2a_{12}a_{23} & 2a_{12}a_{13} \\ a_{23}a_{11} & a_{23}a_{22} & a_{23}a_{33} & 2a_{23}a_{12} & 2a_{23}a_{23} & 2a_{23}a_{13} \\ a_{13}a_{11} & a_{13}a_{22} & a_{13}a_{33} & 2a_{13}a_{12} & 2a_{13}a_{23} & 2a_{13}a_{13} \end{bmatrix} , \tag{3.3}$$

which we handle in VOIGT notation as given in (3.1) – (3.3), respectively.

3.2 Selected Constitutive Models

We exemplary concentrate on two different types of constitutive models to describe the principal strategies and techniques for the implementation. For simplicity, here we reduce to *time independent* problems, which enables us to neglect any discussion on discritizations in time. Nevertheless, we also consider just *isothermal* problems.

3.2.1 Hyperelasticity

Firstly, we have a look on hyperelastic material behavior, which is considered if the stress response is solely given by the derivation of a potential ψ (known as *free energy density*) with respect to the conjugated deformation measure. Again, we refer to the very detailed illustrations in HOLZAPFEL [2000].

Nearly Incompressible Neo–Hooke Model

One of the simplest hyperelastic material models — often used for simple rubber representation and similar material behavior like soft tissues in biomechanical applications — is given by a *free energy density function* of the typical neo–Hookeian type form

$$\psi_{NH} = \underbrace{\frac{1}{2}\mu\,(\bar{I}_1 - 3)}_{\psi_{NH\,iso}} + \underbrace{\frac{1}{2}\,K\,(J-1)^2}_{\psi_{NH\,vol}}\,, \tag{3.4}$$

where we already split ψ_{NH} off into a purely isochoric part $\psi_{NH\,iso}$ and a purely volumetric part $\psi_{NH\,vol}$. Hereby, the isochoric part is driven by $\bar{I}_1$, the first invariant of the modified right Cauchy–Green deformation tensor $\bar{\mathbf{C}} = J^{-2/3}\mathbf{C}$, see Sec. 2.3.1, and the volumetric part is just given as function of the third invariant of the deformation gradient $J = \det(\mathbf{F}) = \sqrt{I_3(\mathbf{C})}$, respectively. Here, nonlinearity is given by the nonlinear deformation measure respecting for finite strains.

In addition, the function ψ_{NH} is determined by only two free material parameters $\mu = G$ and K, which can be clearly identified as *shear modulus* and *compression modulus*, respectively, due to the additive representation in (3.4).

For the simulation of e.g. rubber materials, which are known to react with no volumetric deformation, the part $\psi_{NH\,vol}$ can be seen as incompressibility constraint. In such cases the compression modulus K penalizes the material response and is in the order of $K \cong 2000\,G$ for typical rubber materials.

From the representation (3.4) the stress response in terms of the second Piola–Kirchhoff stress is given by

$$\mathbf{S} = 2\frac{\partial \psi_{NH}(\mathbf{C})}{\partial \mathbf{C}} = \mathbf{S}_{iso} + \mathbf{S}_{vol} = 2\left[\frac{\partial \psi_{NHiso}(\bar{\mathbf{C}})}{\partial \mathbf{C}} + \frac{\partial \psi_{NHvol}(J)}{\partial \mathbf{C}}\right]. \tag{3.5}$$

The belonging modulus can be obtained after some costly manipulations by

$$\mathbb{C} = 2\,\frac{\partial \mathbf{S}(\mathbf{C})}{\partial \mathbf{C}} = \mathbb{C}_{iso} + \mathbb{C}_{vol} = 2\,\frac{\partial \mathbf{S}_{iso}}{\partial \mathbf{C}} + 2\,\frac{\partial \mathbf{S}_{vol}}{\partial \mathbf{C}} \tag{3.6}$$

with

$$\mathbb{C}_{iso} = -\frac{2}{3}\,J^{-2/3}\mu\left[(\mathbf{C}^{-1}\otimes\mathbf{I} + \mathbf{I}\otimes\mathbf{C}^{-1}) - \frac{1}{3}\mathrm{tr}(\mathbf{C})\mathbf{C}^{-1}\otimes\mathbf{C}^{-1} + \mathrm{tr}(\mathbf{C})\frac{\partial \mathbf{C}^{-1}}{\partial \mathbf{C}}\right]$$

and

$$\mathbb{C}_{vol} = KJ\left[(2J-1)\,\mathbf{C}^{-1}\otimes\mathbf{C}^{-1} + 2(J-1)\,\frac{\partial \mathbf{C}^{-1}}{\partial \mathbf{C}}\right]. \tag{3.7}$$

Here, again the decoupled form as in (3.4) is visible, which leads to a representation in mainly two term depending on the parameters G and K and the deformation due to $\mathbf{C}$ and $J = \det(\mathbf{F})$.

For more details, we refer to Holzapfel [2000] and to Miehe [1994], especially for a transformation onto the actual configuration. This representation of a forth order tensor can be treated as 6×6 array as given before, where our preliminaries of the Voigt notation are applied, see Sec. 2.3.3, and — at last — it remains symmetric because of the nature of the constitutive framework. A more enhanced model for rubber elasticity is the well-known Yeoh model with

$$\psi_{Yeoh} = c_1 \,(\bar{I}_1 - 3) + c_2 \,(\bar{I}_1 - 3)^2 + c_3 \,(\bar{I}_1 - 3)^3 + \frac{1}{2} K \,(J-1)^2 \qquad (3.8)$$

as strain energy density function incorporating three material parameters c_1, c_2 and c_3. This model is able to represent the typical upturn behavior in rubber materials for elongated stretches. In Fig. 3.1 we give data of a typical elastomer for a one–axial tension test.

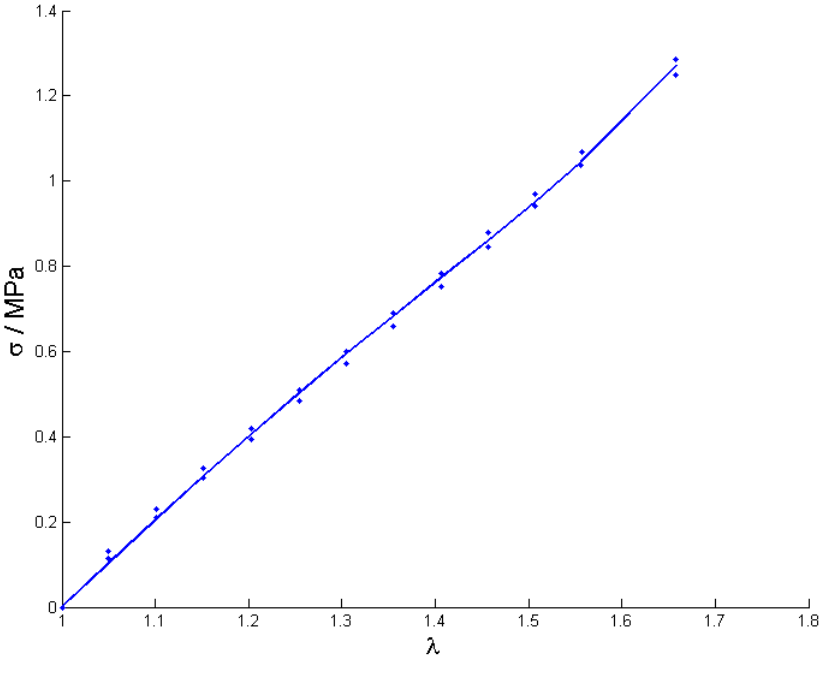

Fig. 3.1 Example of one–axial tension test for an elastomer

3.2.2 Parameter Calibration

The formulation of the YEOH model in the one–axial case with

$$\mathbf{F} = \begin{bmatrix} \lambda & 0 & 0 \\ 0 & \frac{1}{\sqrt{\lambda}} & 0 \\ 0 & 0 & \frac{1}{\sqrt{\lambda}} \end{bmatrix} \tag{3.9}$$

and the additional condition of vanishing stresses perpendicular to the tension direction, namely $t_2 = t_3 \equiv 0$ leads to the one–axial formulation

$$t_{Yeoh}^{1ax} = [2c_1 + 4c_2\,(I_1^{1ax} - 3) + 6c_3\,(I_1^{1ax} - 3)^2]\,(\lambda^2 - \frac{1}{\lambda}) \tag{3.10}$$

for the CAUCHY stress, see (2.27), with the first invariant of the deformation, see (2.19), for the one–axial case $I_1^{1ax} = \lambda^2 - \frac{2}{\lambda}$.

Here, we demonstrate a parameter calibration procedure for the three parameters c_1, c_2 and c_3 by a least–square–fit

```
% init
param = [ 1.0 1.0 1.0 ]

anz_param = length(param);
sprintf('number of parameter: %3i',anz_param)

data = dlmread('1ax.dat');
data = sort(data,1);

sig_mess = data(:,2);
lambda = data(:,1) + ones(length(sig_mess),1);

for i=1:length(lambda)
  sig_mess_CAUCHY(i,1) = sig_mess(i) * lambda(i);
end %i

% minimization
optset = optimset('MaxFunEvals',1.0e6,'TolX',1.0e-8, ...
                                  'Display','iter');
param = fminsearch(@(param) minimiz(param,lambda, ...
                    sig_mess_CAUCHY),param,optset);
```

calling

```
function dist = minimiz(param,lambda,sig_mess_CAUCHY)
sig_yeoh = yeoh(param,lambda);
% Residuum
dist = sig_yeoh - sig_mess_CAUCHY;
```

```
  dist = norm(dist,'fro');
  return
```

calling

```
  function sig_yeoh = yeoh(param,lambda)
  anz_data = length(lambda);
  sig_yeoh = zeros(anz_data,1);
  for i=1:anz_data
    I1_1ax = lambda(i)^2 + 2/lambda(i);
    sig_yeoh(i) = (2*param(1)+4*param(2)*(I1_1ax-3)+ ...
                              6*param(3)*(I1_1ax-3)^2)* ...
                             (lambda(i)^2-1/lambda(i));
  end %i
  return
```

which results for e.g.

```
  0.000   0.000
  0.050   0.125
  0.101   0.210
  0.203   0.349
  0.305   0.459
  0.406   0.557
  0.507   0.644
  0.658   0.775
  0.101   0.190
  0.203   0.328
  0.305   0.438
  0.406   0.536
  0.507   0.624
  0.658   0.754
```

in

```
  param =
      0.3428   -0.0640    0.0277
```

for c_1, c_2 and c_3.

3.2.3 Inelastic Behavior

As second class of typical material modeling we have a look on one possible treatment of (metal) plasticity in the framework of time–independent inelasticity, where *time* plays the role of an identifier to distinguish presence from history events; we symbolize that *scaled time* by τ. In that sense, one often argues with respect to the material behavior — which is always *viscous* by nature — that we just treat very slow effects, so that they are understood as limiting case for $t \to \infty$.

Classical J_2–Plasticity

Kinematics

The framework of multiplicative elastoplasticity is used. Its kinematic key assumption is the multiplicative split of the deformation gradient, which maps material points $\mathbf{X}$ onto the current configuration $\mathbf{x}$,

$$\mathbf{F} := \partial \mathbf{x} / \partial \mathbf{X} = \mathbf{F}_e \cdot \mathbf{F}_p \tag{3.11}$$

into an elastic and an inelastic part, providing the basis of a geometrically exact theory and avoiding linearization of any measure of deformation. Note that

$$\mathrm{d}\hat{\mathbf{x}} = \mathbf{F}_p \cdot \mathrm{d}\mathbf{X} = \mathbf{F}_e^{-1} \cdot \mathrm{d}\mathbf{x} \tag{3.12}$$

introduces a so–called *intermediate configuration*, which quantities are labeled by $\hat{(\bullet)}$. As a further advantage, fast and numerically stable iterative algorithms, proposed and described in Simo [1992], can be used. In the following, only a brief summary of the integration algorithm for a time step $[t_n; t_{n+1}]$ in the context of a FE–implementation is given. Note that in the following the index $n+1$ is suppressed for brevity if misunderstanding is unlikely to occur.

The essential aspect of the multiplicative decomposition is the resulting additive structure of the current logarithmic principal strains within the return mapping scheme as $\boldsymbol{\epsilon}^e = \boldsymbol{\epsilon}^{tr} - \Delta\boldsymbol{\epsilon}^p$. Here, $\boldsymbol{\epsilon}^e$ and $\boldsymbol{\epsilon}^{tr}$ stand for a vector representation with the components $\epsilon_i^e = \ln \mu_i^e$ and $\epsilon_i^{tr} = \ln \mu_i^{tr}$, respectively, strictly connected with the spectral decomposition of the elastic left Cauchy–Green tensor.

The elastic left Cauchy–Green tensor can be specified with the multiplicative decomposition (3.11) as

$$\mathbf{b}_e = \mathbf{F}_e \cdot \mathbf{F}_e^{\mathrm{T}} = \mathbf{F} \cdot \mathbf{C}_p^{-1} \cdot \mathbf{F}^{\mathrm{T}} \ , \tag{3.13}$$

where the superscripts "-1" and "T" denote the inverse and the transpose of a tensor, respectively. That relation clearly shows the "connection" between the elastic and inelastic deformation measure by the occurrence of the inelastic right Cauchy–Green tensor $\mathbf{C}_p = \mathbf{F}_p^{\mathrm{T}} \cdot \mathbf{F}_p$. By means of the relative deformation gradient, see Simo [1992] and Tsakmakis & Willuweit [2003],

$$\Delta\mathbf{F} = \partial \mathbf{x}_{n+1} / \partial \mathbf{x}_n = \mathbf{F}_{n+1} \cdot \mathbf{F}_n^{-1} \ , \tag{3.14}$$

which relates the current configuration $\mathbf{x}_{n+1}$ to the configuration belonging to the previous time step at t_n, an elastic *trial*–state $\mathbf{b}_e^{tr} = \Delta\mathbf{F} \cdot \mathbf{b}_n \cdot \Delta\mathbf{F}^{\mathrm{T}}$ is calculated for the current configuration with frozen internal variables at state t_n.

In the considered case of isotropy, $\mathbf{b}_e$ commutes with $\boldsymbol{\tau}$, see Reese & Wriggers [1997], Simo [1992]. We assume to fix the principal axes of $\mathbf{b}_e$

during the return mapping scheme described in the previous section, so that the spectral decomposition

$$\mathbf{b}_e^{tr} = \sum_{i=1}^{3} \mu_i^{tr\,2}\, \mathbf{n}_i^{tr} \otimes \mathbf{n}_i^{tr} \tag{3.15}$$

is given and the eigenvectors $\mathbf{n}_i^{tr}$ can also be used to compose the stress tensor $\boldsymbol{\tau} = \sum_{i=1}^{3} \tau_i\, \mathbf{n}_i^{tr} \otimes \mathbf{n}_i^{tr}$. That motivates the evaluation of the constitutive equations given in the previous section in principal axes, which means additionally a time saving compared to an evaluation of all six (symmetric) tensor components.

Furthermore, for the elastic part of the material description any hyperelastic behavior can be used.

Remark

The general concept of LIE time derivative $\mathcal{L}_{\mathbf{v}}(\bullet)$ characterizing the change of a spatial field in the direction of the vector $\mathbf{v}$ and known to yield objective spatial fields, see HOLZAPFEL [2000], leads in this case to the OLDROYD rate of the elastic left CAUCHY–GREEN tensor

$$\mathcal{L}_{\mathbf{v}}\mathbf{b}_e = \overset{\triangledown}{\mathbf{b}}_e = \dot{\mathbf{b}}_e - \mathbf{l}\cdot\mathbf{b}_e - \mathbf{b}_e\cdot\mathbf{l}^{\mathrm{T}} \tag{3.16}$$

where $\dot{\mathbf{b}}_e$ denotes the material time derivative and $\mathbf{l} = \operatorname{grad}\dot{\mathbf{x}} = \dot{\mathbf{F}}\cdot\mathbf{F}^{-1}$ the spatial velocity gradient. In this case $\mathbf{v}$ is identified as velocity vector $\mathbf{v} = \dot{\mathbf{x}} = \partial\mathbf{x}/\partial t$. The decomposition of $\mathbf{l} = \mathbf{d} + \mathbf{w}$ in its symmetric part $\mathbf{d} = \operatorname{sym}(\mathbf{l}) = \frac{1}{2}(\mathbf{l} + \mathbf{l}^{\mathrm{T}})$ and its antimetric part $\mathbf{w}$, known as spin tensor, respectively, plays a crucial role in the definition of the inelastic flow rule. Some basic algebraic manipulations let us also obtain the expressions in (3.16) as

$$\mathcal{L}_{\mathbf{v}}\mathbf{b}_e = -2\mathbf{F}_e\cdot\operatorname{sym}(\hat{\mathbf{l}}_p)\cdot\mathbf{F}_e^{\mathrm{T}} = -2\operatorname{sym}(\mathbf{l}_p\cdot\mathbf{b}_e)\ , \tag{3.17}$$

where $\hat{\mathbf{l}}_p$ is defined by $\hat{\mathbf{l}}_p = \dot{\mathbf{F}}_p\cdot\mathbf{F}_p^{-1}$ acting on the intermediate configuration. Please note, that we do *not* make any assumption concerning the antimetric part $\mathbf{w}$ of $\mathbf{l}$. Because of the restriction to isotropic material behavior, the focus is just directed to the symmetric part $\mathbf{d}$ of $\mathbf{l}$. So, the additive decomposition $\mathbf{d} = \mathbf{d}_e + \mathbf{d}_p$ results from the multiplicative decomposition $\mathbf{F} = \mathbf{F}_e\cdot\mathbf{F}_p$.

The definition of the associated flow rule in that finite strain regime as

$$\mathbf{d}_p := \gamma\,\frac{\partial\Phi}{\partial\boldsymbol{\tau}} \tag{3.18}$$

enables with (3.16) and (3.17) the formulation

$$\mathcal{L}_{\mathbf{v}}\mathbf{b}_e = -2\,\mathrm{sym}(\gamma\,\frac{\partial \Phi}{\partial \boldsymbol{\tau}} \cdot \mathbf{b}_e)\ , \tag{3.19}$$

where γ here is known as "consistency parameter", see SIMO [1992].

Stress Representation

Following ARAVAS [1987], we write the KIRCHHOFF stress tensor $\boldsymbol{\tau}$ as the weighted CAUCHY stress tensor as

$$\boldsymbol{\tau} = J\,\boldsymbol{\sigma} = -p^{\tau}\mathbf{I} + 2/3\,q^{\tau}\hat{\mathbf{n}}\ , \tag{3.20}$$

where $J := \det \mathbf{F}$ is the determinant of the deformation gradient. The scalar $p^{\tau} = -\tau_{ij}\delta_{ij}/3$ defines the hydrostatic pressure, $q^{\tau} = \sqrt{3/2\,t_{ij}t_{ij}}$ is the equivalent KIRCHHOFF stress and $t_{ij} = \tau_{ij} + p^{\tau}\delta_{ij}$ are the components of the KIRCHHOFF stress deviator[1]. These quantities can also be obtained for the CAUCHY stress tensor, whose deviatoric stress is $\mathbf{s} = \boldsymbol{\sigma} + p^{\sigma}\mathbf{I}$. In this notation, an additional important quantity is the normalized and dimensionless stress deviator

$$\hat{\mathbf{n}} = 3/(2q^{\tau})\mathbf{t} = 3/(2q^{\sigma})\mathbf{s}\ . \tag{3.21}$$

The second order unit tensor $\mathbf{I}$ is defined as the KRONECKER symbol by its components δ_{ij} in the Cartesian frame.

Internal System of Equations

Analogous to (3.20), the inelastic strain rate can be written as

$$\Delta\boldsymbol{\epsilon}^{p} = \frac{1}{3}\Delta\varepsilon_p\mathbf{I} + \Delta\varepsilon_q\hat{\mathbf{n}}\ , \tag{3.22}$$

where $\Delta\varepsilon_p$ and $\Delta\varepsilon_q$ describe scalar rate quantities which are defined below. Note, that again the dimensionless tensor quantities $\mathbf{I}$ and $\hat{\mathbf{n}}$ are used in this notation.

The two scalar valued quantities $\Delta\varepsilon_p$ and $\Delta\varepsilon_q$ are given with (3.18) as

$$\Delta\varepsilon_p = -\gamma\left(\frac{\partial \Phi}{\partial p}\right) \qquad \text{and} \qquad \Delta\varepsilon_q = \gamma\left(\frac{\partial \Phi}{\partial q}\right)\ . \tag{3.23}$$

The algebraic elimination[2] of γ in (3.23) leads to

$$\Delta\varepsilon_p\left(\frac{\partial \Phi}{\partial q}\right) + \Delta\varepsilon_q\left(\frac{\partial \Phi}{\partial p}\right) = 0\ . \tag{3.24}$$

[1] EINSTEIN summation: $i, j = 1, 2, 3$.

[2] Please note: Here, we consider the very general case, where the plastic strain rate is pressure dependent, which allows that algebraic manipulation. Otherwise, the "consistency parameter" γ has to be determined instead of $\Delta\varepsilon_p$. In that case, we obtain $\Delta s = \sqrt{2/3}\,\Delta\varepsilon_q$ as algorithmic counterpart to $\dot{s}$.

With that and a yield surface of the type

$$\Phi(p,\, q,\, \varepsilon_M^{pl},\, f) = 0 \tag{3.25}$$

the internal set of equations is completed by an isotropic hardening rule of the form

$$k_{iso} = k_{iso}(s)\ , \tag{3.26}$$

where s is the plastic arc–length and its increment $\dot{s} \equiv \gamma$. If the condition $\Phi \leq 0$ (see (3.25)) is fulfilled by the current stress state $\boldsymbol{\tau}$, this state is possible and it is the solution. If, on the other hand, $\Phi \leq 0$ is violated by the trial–state, the trial stresses must be projected back on the yield surface $\Phi = 0$ in an additional step, often called "exponential return mapping". In that case, $\mathbf{x} = \mathbf{x}_{n+1}$ is fixed and (3.19) results in

$$\overset{\triangledown}{\mathbf{b}}_e = \dot{\mathbf{b}}_e = -2\gamma\, \mathrm{sym}(\frac{\partial \Phi}{\partial \boldsymbol{\tau}} \cdot \mathbf{b}_e)\ , \tag{3.27}$$

with $\mathbf{l} \equiv \mathbf{0}$. The solution of the first order differential equation (3.27) is given by

$$\mathbf{b}_{e,\,n+1} = \sum_{i=1}^{3} \underbrace{\exp\left[2\epsilon_i^e\right]}_{\mu_i^{e\,2}}\, \mathbf{n}_{(i)\,n+1}^{tr} \otimes \mathbf{n}_{(i)\,n+1}^{tr}\ , \tag{3.28}$$

where the elastic logarithmic strains $\boldsymbol{\epsilon}^e$ are obtained in principal axes, see (3.29) below, so that $\mathbf{b}_{e,\,n+1}$ is known and $\mathbf{C}_p^{-1} = \mathbf{F}^{-1} \cdot \mathbf{b}_{e,\,n+1} \cdot \mathbf{F}^{-\mathrm{T}}$ can be stored as *history variable* for the next time step.

HISTORY–Storage

With these results of the integration procedure above for $\Delta\boldsymbol{\epsilon}_{n+1}^p$, the actual elastic left Cauchy–Green tensor is computed by

$$\boldsymbol{\epsilon}_{n+1}^e = \boldsymbol{\epsilon}^{tr} - \Delta\boldsymbol{\epsilon}_{n+1}^p \tag{3.29}$$

and (3.28) in reversal of the spectral decomposition (3.15), see Reese & Wriggers [1997].

Rousselier **Damage Model**

See Baaser & Tvergaard [2003], where the implementation of a damage model in the extended sense of Sec. 3.2.3 is demonstrated and successfully applied.

4

FEM Implementation within MATLAB

Starting this chapter, where we describe the realization of `DAEdalon` within the MATLAB environment, we quickly itemize the main features of MATLAB operating on vector and matrix representations, respectively. This list can not substitute a deeper study of the printed MATLAB manuals or any tutorial obtained from the internet or even the built–in `help` function. This fact is recognized very quickly: MATLAB can not be learned by reading, but only by programming in the environment !

4.1 MATLAB Basics

- A "vector" in the physical (EUKLIDian) space $\mathbb{E}^3$ is read as column array of the three coefficients a_1, a_2 and a_3 in the form

$$\mathbf{a} := \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} \tag{4.1}$$

  represented and implemented in MATLAB by $\mathtt{a} = [\mathtt{a}_1; \mathtt{a}_2; \mathtt{a}_3]$, see (2.1). With respect to the continuum mechanical background in Chap. 2, we avoid here a further notation of the tensorial basis system and denote all vectors and tensors in the above given (EUKLIDian) space, if nothing else is mentioned.
- In contrast to that, a transposed vector $\mathbf{a}^{\mathrm{T}}$ is given by $\mathtt{a}'$ or directly by $\mathtt{a\_tr} = [\mathtt{a}_1\ \mathtt{a}_2\ \mathtt{a}_3]$ as row type array. Please note, the semicolon ";" as given above forcing a line break are omitted here in order to represent a row vector.
- Consequently, the vector (dot) product, see (2.2), ending in a scalar quantity is given by $\mathtt{c} = \mathtt{a\_tr} * \mathtt{a}$ or $\mathtt{c} = \mathtt{a}' * \mathtt{a}$.
- Further on, a 3×3 array (dyadic) representation is given e.g. by $\mathtt{C} = \mathtt{a} * \mathtt{a}'$, see (2.3).
- MATLAB coding is case sensitive, so that `c` is not equal to `C` !

- In addition, we often make use of the command
  `reshape(C,[num_col num_row]`,
  where `num_col` represents the *new* number of columns and `num_row` the *new* number of rows of the array **C**. Please note, that the total number of entries `num_col*num_row` remains constant during this operation.
- Finally, we want to emphazise the "blockwise" operation and storing technique of MATLAB, where one can operate and manipulate on single parts of a (large) array. We make use of that functionality especially during the assembling procedure, see Sec. 4.4.1, or directly in `stiffness_func.m`, which contains the command line
  `k(pos_vec,pos_vec) = k(pos_vec,pos_vec) + k_elem;`
  adding directly the element contributions into the global system array.

4.2 Structure of DAEdalon

Since the first steps in the development of `DAEdalon`, in the following of the year 2000, many contributions and details are added — and are ongoing added actually, so that the aim of the whole booklet is a representation of the main and basic structure and the architecture of the system behind.

Principally, the main skeleton of `DAEdalon` is given schematically in Fig. 4.1, which represents the m–file structure of the system. The initialization of the program and loading of the actual problem is activated by `dae` and `lprob`, respectively. The time counter is set by the `time` command. At the that state, the capabilities described in Sec. 4.5 can be obtained to check or control the defined

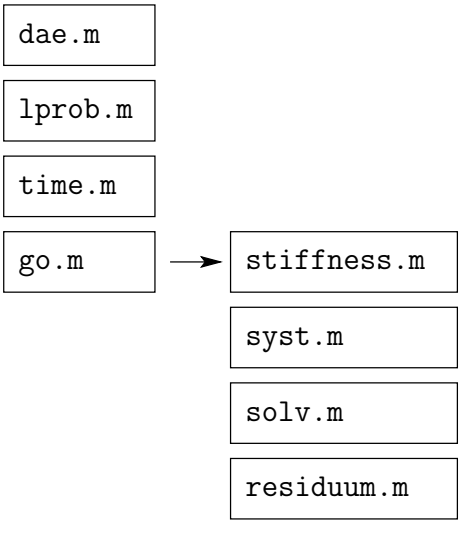

Fig. 4.1 Schematic Carry–Out of a `DAEdalon` Analysis

problem. The numerical solution is activated afterwards by `go`, which is just an assembly of the in Fig. 4.1 visualized commands. Obviously, that is the core part of `DAEdalon` representing a development tool for FE.

Please, feel free to add your own developments and ideas within that given structure ...

4.3 Preprocessing

`DAEdalon` requests at least for the six input data files `node.inp`, `el.inp`, `mat1.inp`, `geom.inp`, `displ.inp` and `force.inp` as already explained in Sec. 1.5. As mentioned before, the focus of `DAEdalon` is the solution part of the FEM, for which we decide to neglect sophisticated preprocessing tools available in many CAD– and special preprocessing software–packages in meantime.

4.3.1 Structure of Input-Files and Processing

`geom.inp`
  line 3 — `nummat`
  line 5 — `ndf`
`node.inp`
  `numnp`
  `ndm`
`el.inp`
  `numel`
  `nel`

After reading that basic input data files, the system computes and initializes automatically the global variables and array dimensions, e. g. *total number of degrees of freedom* `gesdof = ndf*numnp` giving the size of the solution vector $\mathbf{u}$ and its increment d$\mathbf{u}$, the size of the *right hand side* `rhs`.

4.4 Numerical Solution

The main task of the numerical implementation is the solution of (2.52) with respect to the solution vector $\mathbf{u}$, which can be realized in `DAEdalon` along with Fig. 4.1.

4.4.1 Assembly Procedure — The Global System

The two core commands on the global level of FE analysis within `DAEdalon` are the scripts `stiffness.m` and `syst.m`, which assemble the global system matrix and the *right hand side* vector and consider, finally, for the given boundary conditions.

In detail, the `stiffness.m` script loops over all given elements in all given material sets and collect the element information, i. e. the element stiffness `kmat` given in (2.53) and the element *right hand side* vector `rhs`, see Sec. 2.55. For each element call, the necessary information as actual displacements $\mathbf{u}^e$ or the connected history data is selectively given out from the global fields.

The following is the assembly procedure of the global system, which is in that way the main advantage of doing it within the MATLAB environment. We are able to handle the particular element information as sub–/block–matrix and store it very easily in the global arrays by operations like

```
kmat(1:2, 3:4) = kmat_elem()
```

resulting in the global system matrix $\mathbf{K}$.

Further on, `syst.m` considers for the applied (displacement) boundary conditions and the load array. Here, each given (displacement) boundary condition leads to a reorganization of the global system of equations. Because of its known entry, the respecting information is sorted into the right hand side of the system and the corresponding column and row of `kmat` is filled completely with zero entries, except for the diagonal entry, which is filled with "1". So, the global system remains it original size `numnp * ndf` equations, but the system matrix `kmat` gets not singular and is invertible by that treatment.

4.4.2 NEWTON–*Iteration*

As described in Sec. 2.7.2, we start with (2.52) and the assembled global stiffness resulting from (2.53). Numerically, we solve (2.55), which is given as

```
kmat * du = rhs ,
```

by

```
du = kmat \ rhs .
```

Here, `rhs` represents the discretization of $-g(\hat{\mathbf{u}})$ in vector form. Doing so, we are able to update $\mathbf{u}$ by

```
u = u + du
```

within the iteration loop, see `solv.m`. In these loops we control the quality of the solution by the norm

```
res_norm=sqrt(rhs'*rhs)
```

of resulting right hand side vector `rhs`, see `residuum.m`. The user is free to end the iteration process, if that norm falls below a tolerance of e.g. 10^{-9}, which denotes numerically the *zero*.

4.5 Postprocessing

4.5.1 Mesh Representation, Loads and Boundary Conditions

We provide a number of predefined m–files for the postprocessing and the visualization of the obtained FE results. Of special interest are the commands `mesh0`, `meshx`, `nodenum` and `elnum` for the representation of the mesh state in the undeformed and deformed configuration and the numbering of the nodes and elements, respectively.

In addition, the `forc` and `boun` commands give the applied nodal loads as arrows and the displacement restraints as line at the subjected nodes directed in the given direction. Both commands represent the content of the input data base given in `forc.inp` and `displ.inp`.

4.5.2 Contour-Plots

The treatment and the computation of a requested quantity by the `cont(·)` command is explained in the following. To compute a smooth field p in the sense of a least square fit from e.g. the (discrete) scalar pressure p^h projected to the FE nodes obtaining the nodal values $p^{node} = \sum_I N_I \, p_I^{node} = \mathbf{N}^{\mathrm{T}} \cdot \mathbf{p}^{node}$, see also (2.39), we consider the minimization

$$\int_{\mathcal{B}} \frac{1}{2}(p - p^h)^2 \mathrm{d}v \rightarrow min \rightsquigarrow \int_{\mathcal{B}} \delta p \, (p - p^h) \mathrm{d}v = 0 \; . \tag{4.2}$$

These quantities can be prepared elementwise and stored there as

$$\underbrace{\mathsf{A} \int_{\mathcal{B}} \mathbf{N} \cdot \mathbf{N}^{\mathrm{T}} \mathrm{d}v}_{\text{nominator}} \; \mathbf{p}^{node} = \underbrace{\mathsf{A} \int_{\mathcal{B}} \mathbf{N} \cdot p^h \mathrm{d}v}_{\text{numerator}} \; , \tag{4.3}$$

which can be solved very easy for $\mathbf{p}^{node}$. So, the nodal contribution $\mathbf{p}^{node}$ from each element is obtained by a assembly procedure over all elements very analogous to the construction of the element stiffness matrix and the load vector. For that reason, we compute the numerator and the nominator contribution out of (4.3) in the same loop within the element formulation and store them in the arrays `cont_zaehler` and `cont_nenner`, respectively.

4.5.3 GUI — Realization of a Graphical Environment

The above given commands — and some additional features of `DAEdalon` — can also be obtained and executed by a **G**raphical **U**ser **I**nterface activated

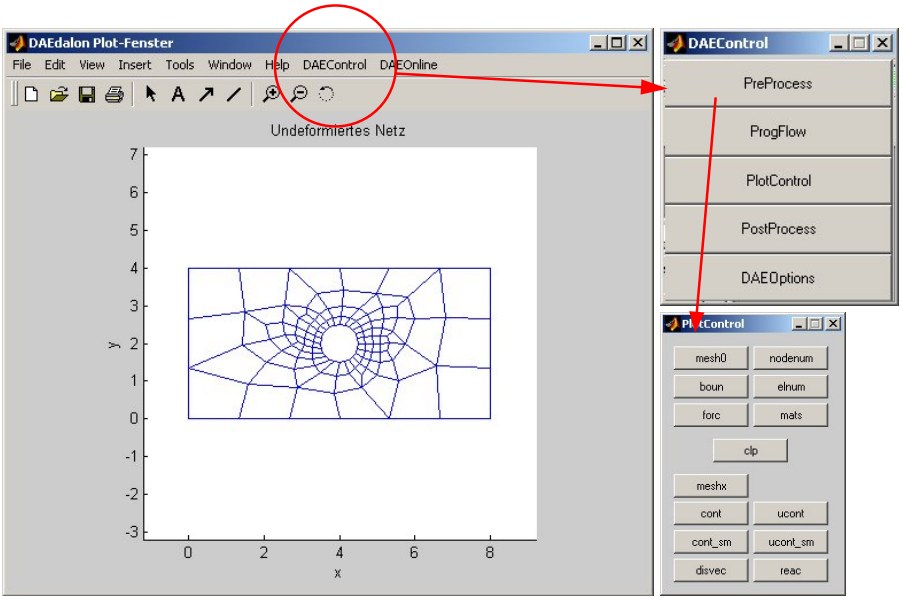

Fig. 4.2 The `DAEdalon` GUI with Example of Sec. 1.2 in parallel to the MATLAB Plot Window. The additional windows are activated by the `DAEControl` Button.

by the `DAE Control Button` in the plot window as is given exemplary in Fig. 4.2. Nevertheless, again the user is able to add his own ideas and to modify that given version of the GUI.

4.6 Writing Extensions for `DAEdalon`

For a demonstration of the coding of a finite element formulation on principle, we refer to `elem4.m`, which has already been considered in Sec. 1.2 for a first introduction and show a formulation considering for rotational symmetry and the EAS formulation of that element enabling us to model highly incompressible material behavior. In addition, we describe the interface for implementing material models and give the formulation of a general purpose 2– or 3–d truss element with nodes.

4.6.1 Continuum Element `elem4.m` *with 4–Nodes in Plane Strain*

The implementation of a finite element within `DAEdalon` is given as MATLAB function in the form

```
function [k_elem, r_elem, cont_zaehler, cont_nenner, ...
                hist_new_elem, hist_user_elem] = ...
    elem4(isw, nel, ndf, contvar, mat_name, mat_par, x,u_elem,
                hist_old_elem, hist_user_elem)
```

where the nodal positions `x` and the nodal displacements `u_elem`, the material parameters `mat_par` and the history data are given besides the number of nodes `nel` and the number of degrees of freedom per node `ndf`.

Generally, one has to define the element stiffness in form of the matrix

$$\mathbf{K}^{elmt} := \mathbf{K}_{mat} + \mathbf{K}_{geom} \ , \tag{4.4}$$

see (2.53), and the element residual or load vector

$$\mathbf{r}^{elmt} := \int_v \mathbf{B}^{\mathrm{T}} \cdot \check{\boldsymbol{\sigma}} \, \mathrm{d}v \ , \tag{4.5}$$

see (2.52), respectively. In Sec. 2.6.2 we have seen a description of the **B**–matrix.

This is realized e.g. in the above mentioned coding of `elem4.m` as

```
for aktgp=1:numgp
  ...
  k_elem = k_elem + b' * D_mat * b * dv;
  r_elem = r_elem + b' * sig * dv;
  ...
end  % Loop aktgp
```

as the loop over the integration point representing the volume integration as shown in Sec. 2.6.4. In addition, that loop contains the computation as well of the resulting stress vector $\check{\boldsymbol{\sigma}}$ including the material modulus $\mathbb{D}$ in the form of `D_mat` as the **B**–matrix as array of the derivatives of the shape functions.

The `cont` variables are used for the contour plot functions and the mapping of the data therein and define the nominator and the numerator in (4.3).

4.6.2 Formulation for Rotational Symmetry of `elem4.m`

The above presented prototype element formulation of a 4–noded continuum element can also be used for axisymmetric descriptions of rotational bodies as given in Fig. 4.3. Just small modifications are required in order to describe an axisymmetric 3–dimensional body under rotational load and deformation conditions. For that, we apply the following cylindrical coordinate system with the labels x_1 for the radial, x_2 for the axial direction and x_3 for the tangential or circumferential direction, respectively. As given in that formulation, the axial direction has always to be adjusted to x_2 as rotational axis.

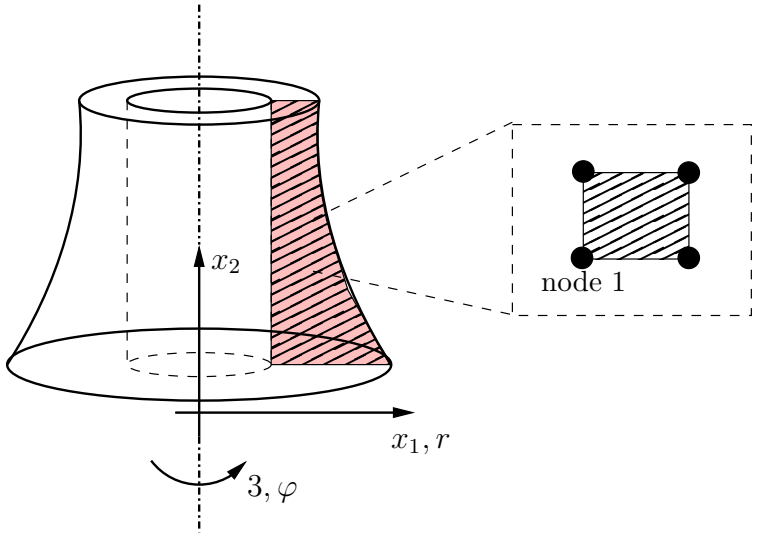

Fig. 4.3 Axisymmetric Formulation of `elem4.m`: Cross Section of an Arbitrary Rotational Symmetric Body and Labeling of the used Directions. Discretization of the Cross Section by 4–Noded Elements.

The given element formulation is based to the deformation gradient for that assumptions as

$$\mathbf{F} = \begin{bmatrix} 1 + \frac{\partial u_r}{\partial x_1} & \frac{\partial u_r}{\partial x_2} & 0 \\ \frac{\partial u_2}{\partial x_1} & 1 + \frac{\partial u_2}{\partial x_2} & 0 \\ 0 & 0 & \frac{r}{R} \end{bmatrix} , \tag{4.6}$$

which is the basis of the strain computation in the constitutive model.

Following this notation, we obtain the stress response in the same order, such that the first index gives the radial component and the second one the axial component. As in the plane strain element formulation in Sec. 4.6.1, we store the shear response σ_{12} in the third component. In Sec. 5.2 we give an application of that formulation.

4.6.3 EAS *Expansion of* `elem4.m`

In order to overcome some disadvantages of the presented *prototype* element formulation given in `elem4.m`, many modifications of that element are well known in literature, see e.g. BELYTSCHKO *et al.* [1984] for a first proposal. The main disadvantage of the element is given by its too stiff response ("*locking behavior*"), especially in mechanical cases which are dominated by bending or whose material behavior is going to the limit case of incompressibility.

The simplest and most *intuitive* idea was to modify the integration scheme of the element especially in the volumetric part of the stress response from a 2×2 version as given in Sec. 2.6.4 into an 1–point scheme, which soften the element behavior. Nevertheless, this modification raises in an additional lack of that formulation: In unprofitable loading cases the element may response with *physically not motivated* deformation modes, which are denoted due to the resulting shapes as *hourglassing*.

Today, the most attractive and modern version of element formulations is known as *enhanced assumed strain (EAS)* formulation, which are based mainly on Simo & Armero [1992] and Andelfinger & Ramm [1993].

This section should describe the main underlying idea of that concept, which is widely used in many element formulation, generally to prevent too stiff element responses. Starting point of such formulations is an enhancement of the variational principle 2.33 into a 3–field functional (Hu–Washizu principle) with an internal energy expression

$$\Pi^{e,int} := \int_{V_e} \frac{1}{2} \boldsymbol{\varepsilon}_h^{\mathrm{T}} \cdot \mathbb{C} \cdot \boldsymbol{\varepsilon}_h - \boldsymbol{\sigma}_h^{\mathrm{T}} \cdot \boldsymbol{\varepsilon}_h + \boldsymbol{\sigma}_h^{\mathrm{T}} \cdot \mathbf{B} \cdot \mathbf{u}_h \, \mathrm{d}V_e \tag{4.7}$$

introducing the *enhanced strains* $\tilde{\boldsymbol{\varepsilon}}_h$ in addition to the (original) *compatible* strains in

$$\boldsymbol{\varepsilon}_h = \underbrace{\mathbf{B} \cdot \mathbf{u}_h}_{compatible} + \underbrace{\tilde{\boldsymbol{\varepsilon}}_h}_{enhanced\ strains} \quad . \tag{4.8}$$

Obviously, this additional and *fictive* strain field may not cause any energy turnover within the element formulation, which delivers the additional condition

$$\int_{V_e} \boldsymbol{\sigma}_h^{\mathrm{T}} \cdot \tilde{\boldsymbol{\varepsilon}}_h \, \mathrm{d}V_e \stackrel{!}{=} 0 \tag{4.9}$$

in an elementwise view. In that sense, the introduced strain field $\tilde{\boldsymbol{\varepsilon}}_h$ may be *incompatible* to the mechanical strains and do not give any amount to (4.7).

Furthermore, the FE interpolation of

$$\boldsymbol{\varepsilon}_h = \sum_k M_k \, \alpha_k = \mathbf{M} \cdot \boldsymbol{\alpha} \tag{4.10}$$

in analogy to (1.7) is given by a matrix representation $\mathbf{M}$ with k columns, which is discussed below.

Following the argumentation as in the standard element formulation, leads to a system of equations on the element level of the form

$$\begin{bmatrix} \mathbf{D} & \mathbf{L} \\ \mathbf{L}^{\mathrm{T}} & \mathbf{K}^e \end{bmatrix} \cdot \begin{bmatrix} \boldsymbol{\alpha} \\ \mathbf{u}^e \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{l}^e \end{bmatrix} \tag{4.11}$$

where $\boldsymbol{\alpha}$ denotes here additional degrees of freedom and the block matrices are given by

$$\mathbf{K}^e = \int_{V_e} \mathbf{B}^{\mathrm{T}} \cdot \mathbb{C} \cdot \mathbf{B} \, \mathrm{d}V_e \tag{4.12}$$

as before, see (1.12), and

$$\begin{aligned} \mathbf{D} &= \int_{V_e} \mathbf{M}^{\mathrm{T}} \cdot \mathbb{C} \cdot \mathbf{M} \, \mathrm{d}V_e \ , \\ \mathbf{L} &= \int_{V_e} \mathbf{M}^{\mathrm{T}} \cdot \mathbb{C} \cdot \mathbf{B} \, \mathrm{d}V_e \ . \end{aligned} \tag{4.13}$$

Assuming to know all expressions in (4.11), one obtains a *modified* element stiffness matrix

$$\mathbf{K}^{e,mod} = \mathbf{K}^e - \mathbf{L}^{\mathrm{T}} \cdot \mathbf{D}^{-1} \cdot \mathbf{L} \tag{4.14}$$

by a local condensation of the system on the element level, which is assembled to the global system as described before.

To complete this short introduction into the EAS element formulation, we show the determination of the matrix $\mathbf{M}$ in (4.10), which is described and motivated in detail in ANDELFINGER & RAMM [1993] by a discussion on the deficiencies of the classical element shape functions given in (2.38) with respect to a complete set of polynomial terms. It is shown, that the best approximations can be obtained by setting

$$\mathbf{M} := \frac{\det \mathcal{J}_0}{\det \mathcal{J}} \, \mathbf{T}_0^{-\mathrm{T}} \cdot \mathbf{M}_\xi \ , \tag{4.15}$$

where the index 0 denotes the evaluation of $\mathcal{J}$, see (2.43), and

$$\mathbf{T}_0 = \begin{bmatrix} \mathcal{J}_{0\,11}^2 & \mathcal{J}_{0\,12}^2 & 2\mathcal{J}_{0\,11}\mathcal{J}_{0\,12} \\ \mathcal{J}_{0\,21}^2 & \mathcal{J}_{0\,22}^2 & 2\mathcal{J}_{0\,21}\mathcal{J}_{0\,22} \\ \mathcal{J}_{0\,11}\mathcal{J}_{0\,21} & \mathcal{J}_{0\,12}\mathcal{J}_{0\,22} & \mathcal{J}_{0\,11}\mathcal{J}_{0\,22} + \mathcal{J}_{0\,12}\mathcal{J}_{0\,21} \end{bmatrix} \tag{4.16}$$

in the respective element center at $\xi = \eta = 0$.

Finally, the matrix $\mathbf{M}_\xi$ with additional terms of the shape functions is given for $k = 4$ by

$$\mathbf{M}_\xi = \begin{bmatrix} \xi & 0 & 0 & 0 \\ 0 & \eta & 0 & 0 \\ 0 & 0 & \xi & \eta \end{bmatrix} \ , \tag{4.17}$$

which is shown to be sufficient for the `elem4` formulation. This formalism can expanded easily for $k > 4$ by adding columns to $\mathbf{M}_\xi$ containing additional term as e.g. $\xi\eta$.

Doing so, a very versatile usable element formulation can be realized, which can be applied in many structural mechanics tasks especially those with a

nearly incompressible material response as metal plasticity or in rubber components.

4.6.4 General Truss Element elem10.m with 2 Nodes

In order to formulate this simple finite element, the weak form (2.33) sounds like

$$\int_l A\sigma\,\delta\varepsilon\,\mathrm{d}x - F_1\,\delta u|_{\partial\mathcal{B}_\sigma} = 0\,, \tag{4.18}$$

where A denotes the truss cross section, σ the axial stress and $F = A\,\sigma$ the axial (outer) force loading the truss. As given in Fig. 4.4, the axial displacement u is obtained by the projection of $\mathbf{t}$ and the spatial displacements

$$u = \mathbf{t}\cdot\mathbf{u} = \sum_{I=1}^{2} N_I(\hat{x})\,\mathbf{t}\cdot\mathbf{u}_I\,, \tag{4.19}$$

with the local shape functions

$$N_1(\hat{x}) = 1 - \frac{\hat{x}}{l_e} \quad \text{and} \quad N_2(\hat{x}) = \frac{\hat{x}}{l_e}\,. \tag{4.20}$$

With that, we can easily obtain the local strain as

$$\varepsilon = \frac{\mathrm{d}}{\mathrm{d}\hat{x}}\,u = \sum_{I=1}^{2} \frac{\mathrm{d}N_I(\hat{x})}{\mathrm{d}\hat{x}}\,\mathbf{t}\cdot\mathbf{u}_I = \sum_{I=1}^{2} \mathbf{B}_I\cdot\mathbf{u}_I \tag{4.21}$$

or as given e.g. in (1.7) collecting the derivatives of the element shape functions in a $\mathbf{B}$–matrix, here given by $\mathbf{B}_I = N_{I,\hat{x}}\,\mathbf{t}^{\mathrm{T}}$. In the following, we give

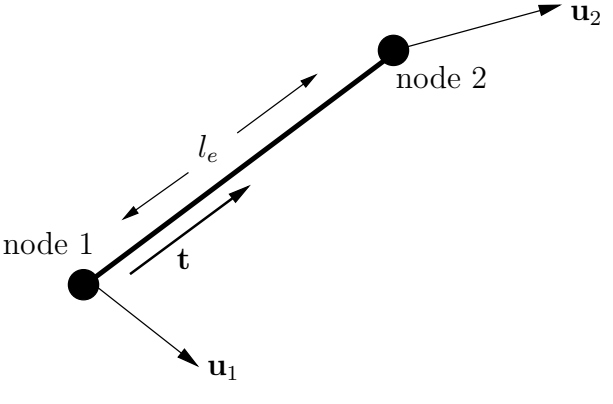

Fig. 4.4 2–Noded Truss Element

the numerical realization of that derivations in a form as implemented in `elem10.m`:

```
% material parameters
E_mod = mat_par(2);
A = mat_par(3);

% normalized axial vector and length
t = x(2,:)' - x(1,:)';
L = norm(t);
t = t/L;

% local displacement
u_axial = (u_elem(2,:) - u_elem(1,:))*t;

% strain
epsilon = u_axial/L;

% stress / HOOKE-model
sig = E_mod*epsilon;

% B-matrix
B = [-t',t']/L;

% element stiffness k_elem
k_elem = B' * E_mod*A * B * L;

% element residual vector
r_elem = B' * sig*A*L;
```

Again, the main results of that MATLAB (element) function is the element stiffness matrix $\mathbf{K}^{elmt}$ in `k_elem` and the element residual vector $\mathbf{r}^{elmt}$ in `r_elem`, respectively.

4.6.5 Material Models

Interested in applying and testing own, new material formulations, one can use likewise material interface functions as given exemplary such as `mat33.m` in the form

```
function [sig,vareps,D_mat,hist_new_gp,hist_user_gp] ...
       = mat33(mat_par,F,hist_old_gp,hist_user_gp)
```

In that sense, please give the strain measure e.g. $\mathbf{E}$ or $\mathbf{e}$, see (2.17) and (2.18), respectively, in `vareps` in VOIGT structure as in (2.29). The material model should compute the conjugated stress response $\mathbf{S}$ or $\boldsymbol{\sigma}$ in the array `sig` and the material modulus $\mathbb{D}$ in `D_mat`, see Sec. 3.1.2.

Here, we focus on the local NEWTON iteration used in Sec. 3.2.3 for the determination of plastic strain increments $\Delta\boldsymbol{\epsilon}^p_{n+1}$:

```
while abs(phi) > 1.0E-12
  dphi = -2.0*mu - 2.0/3.0 * Emod/N * ...
   (Emod/Y0*(alpha+sqrt(2.0/3.0)*gamma)+1.0)^(1.0/N -1.0);
  dgamma = - phi/dphi;
  gamma = gamma + dgamma;
  phi = dev_tau_abs - sqrt(2.0/3.0)*Y0 * (Emod/Y0*(alpha+
    sqrt(2.0/3.0)*gamma)+1.0)^(1.0/N) - 2.0*mu*gamma;
end % Newton-Iteration
```

In the case of e.g. damage models or other highly sophisticated material descriptions, one should use the same interface function to compute the increments of the considered internal variables. As mentioned before, in BAASER & TVERGAARD [2003] exemplary the implementation of the ROUSSELIER damage model is shown, which is treated exactly in that sense.

5

Applications and Examples

5.1 Crane Modeled by 2D–Trusses

In this section we describe the usage of the general purpose 2d– or 3d–truss element as described before in Sec. 4.6.4 for small strain regime by a real–sized engineering example. Obviously, the following model of a building site crane is very simplified, but it shows quite impressively the capacity of the finite element method in order to obtain numerical results for realistic requests. We assume the model of a crane as given in Fig. 5.1, where the structure

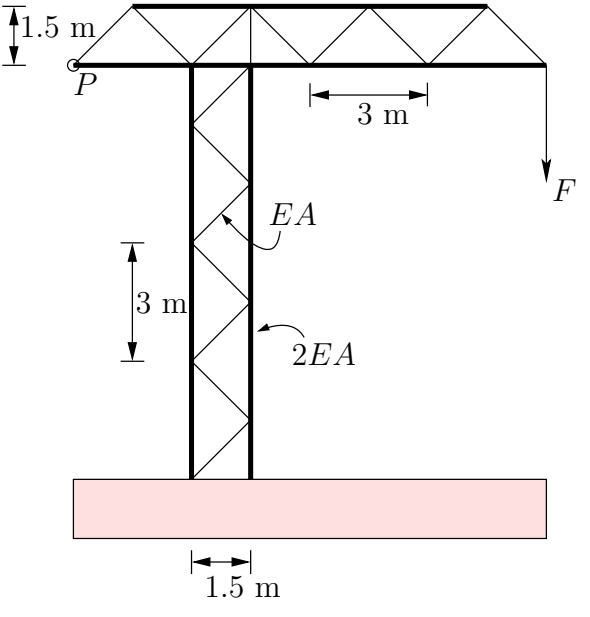

Fig. 5.1 Crane Model by Truss Elements

and the geometrical sizes are given. Further on, we use a YOUNGs modulus of $E = 210000$ MPa for steel and a cross section of the diagonal trusses of $A = 5$ cm^2 and of $2A$ for the main trusses, so that the truss stiffness EA for each segment is given. The given structure is loaded by a force of $F = 8000$ N, while the displacements of point P as given in Fig. 5.1 are requested.

Solution

We realize a numerical solution of the given problem firstly by a discretization of the structure by 2–noded truss elements, where the junctions of the segments represent our finite element nodes and the segments itself the finite (truss) elements. Additionally, due to the different cross sections given above, we define two material sets in the model, which respect for the two different cross sections, see input files `mat1.inp` and `mat2.inp`. The proposed input files for the model are given at `www.DAEdalon.org/Examples/Crane`. In that model, the marked point P is identified with *node 11*, where we want to compute the displacement results. Doing so, following the same procedure as described in the very first example in Sec. 1.2, we are able to have a look at the deformed structure by `meshx` as also given in Fig. 5.2. Getting more details of the nodal solutions, we can type e.g. `dis(11)` resulting in $u_x^P = 23.57$ mm and $u_y^P = 23.11$ mm as the nodal displacements for *node 11*.

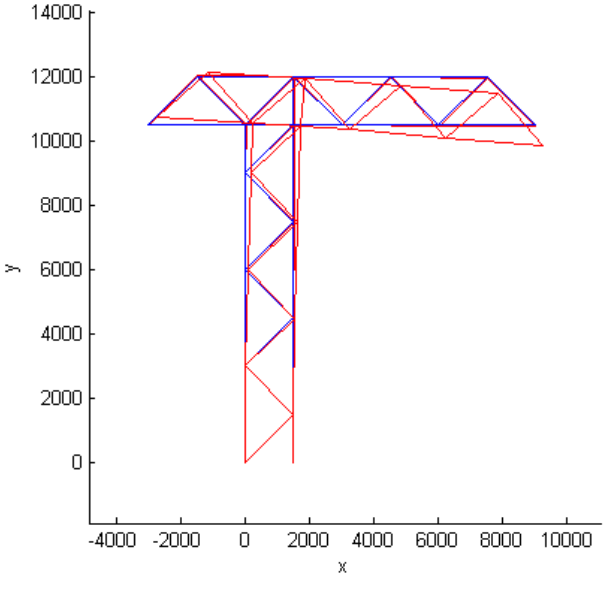

Fig. 5.2 FE model and displacements by `meshx`. Deformations scaled by factor 10 (`defo_scal=10`).

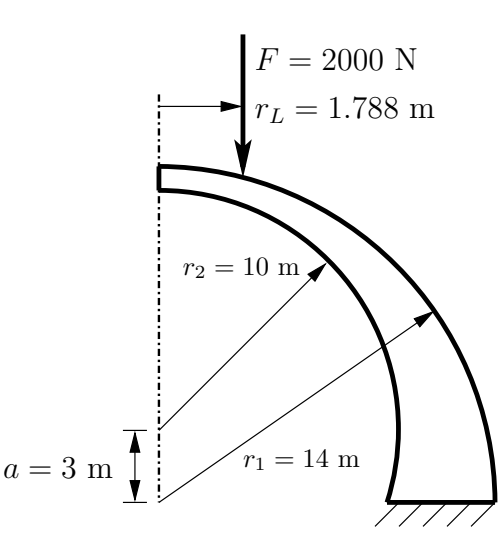

Fig. 5.3 Axisymmetric Concrete Structure under Ring Load **F** with Radius r_L

5.2 Axisymmetric Applications

In order to show an application for the axisymmetric element formulation of Sec. 4.6.2, we consider a dome structure of reinforced concrete, assumed to be homogeneous, isotropic and linearly elastic with YOUNGs modulus $E = 20$ GPa and POISSONs ratio $\nu = 0$ as given in Fig. 5.3. Again, a proposal for the node and element information is given at `www.DAEdalon.org/Examples/Dome`, so that a solution of that task can easily be obtained.

Further on, we want to focus here on the determination of the reaction forces at the mounting of the structure, which is given in more detail including node numbers in Fig. 5.4. As given, we have to add the computed reaction forces in axial–/ 2–direction of nodes 10, 17–20 and 109–112 in a postprocess after solving the linear elastic mechanical problem following the standard procedure shown in Sec. 1.5. At that point, we have a look on the structure of the used (internal) fields, e.g. `rnode` in this case. The matrix `rnode` is of size [`numnp` $\times$ 3], where `numnp` gives the *number of nodes*, while the columns are reserved for *3* dimensions. So, the rows in `rnode` contain the *(reaction) forces* of each node after solving the given boundary value problem. Consequently, not restricted degrees of freedom are filled with 0.0 in `rnode`. In our case, we can obtain the requested mounting loads by adding the second column values of `rnode` of the bounded nodes 10, 17–20 and 109–112, or just by summing up the second column of `rnode` and subtracting the applied load of 2000 N at node 35:

```
sum(rnode(:,2))-rnode(35,2)
```

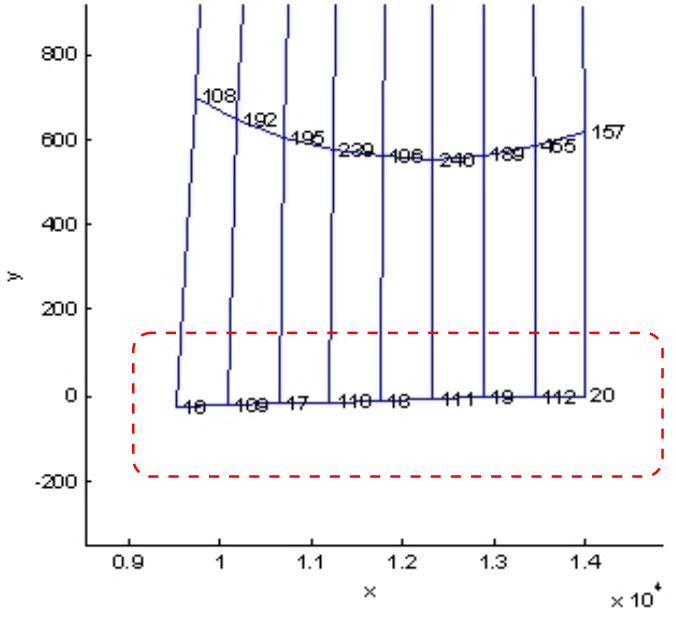

Fig. 5.4 Mounting of Structure with Node Numbers

The resulting sum is according to amount the same as the applied load at the top of the structure, which additionally indicates, that the system is in mechanical equilibrium.

5.3 Crack Tip Simulation

A further, more advanced application of the FEM is given by the analysis of a crack tip field, whose order of magnitude is described by a *crack intensity factor* K_I. We want to address the computation of a given K_I–field in comparison to the well discussed analytical solution, see e.g. Gross & Seelig [2006].

5.3.1 K_I–Field

We describe a crack tip in a plane strain configuration as given in Fig. 5.5 with a radius of $r = 0.1$ mm. As known, this region is loaded by boundary conditions from a K_I–dominated far–field of the form

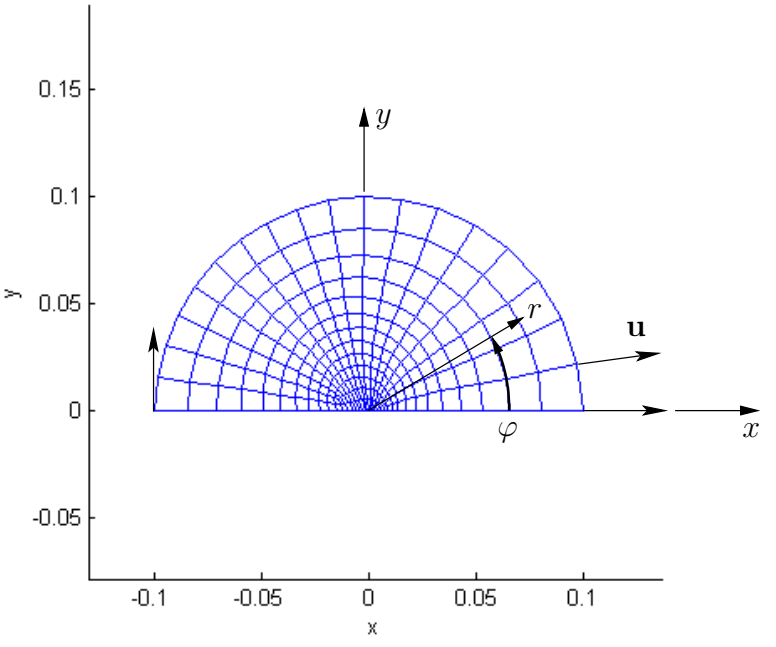

Fig. 5.5 Discretization for K_I–Mode Crack Tip Analysis

$$u_x = \frac{K_I}{2G}\sqrt{\frac{r}{2\pi}}\,(3-4\nu-\cos\varphi)\,\cos\frac{\varphi}{2}$$
$$u_y = \frac{K_I}{2G}\sqrt{\frac{r}{2\pi}}\,(3-4\nu-\cos\varphi)\,\sin\frac{\varphi}{2}\,, \tag{5.1}$$

where we choose a linear elastic material behavior through $E = 210000$ MPa and $\nu = 0.3$ and a loading intensity of $K_I = 10$ MPa$\sqrt{\text{m}}$. The solution of the given problem computed with `DAEdalon` gives a σ_y stress contour as given in Fig. 5.6. As additionally given in textbooks on fracture mechanics depicting the K–concept, one obtain a typical opening

$$u_y^{\pm} = \pm\frac{2\,K_I}{G}\sqrt{\frac{r}{2\pi}}\,(1-\nu) \tag{5.2}$$

of the crack edges for the chosen situation. In Fig. 5.7 we give that analytical result (5.2) of the *crack tip opening displacement* in comparison to the FE solution.

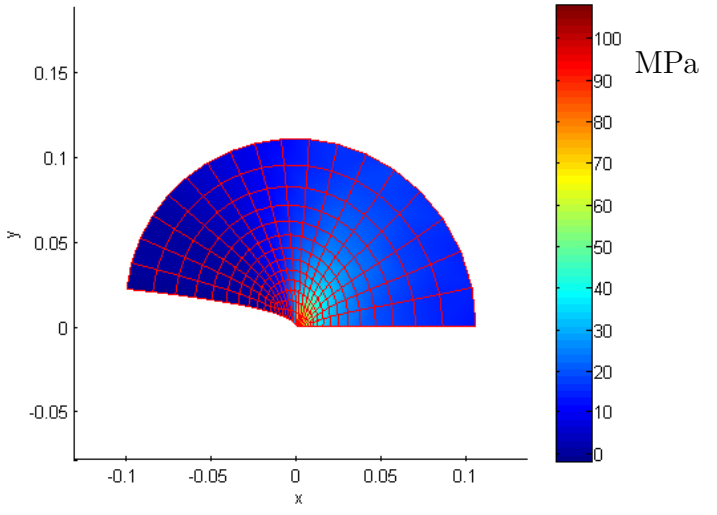

Fig. 5.6 σ_y stress distribution on scaled configuration by `meshx`. Deformations scaled by factor 1000 (`defo_scal=1000`).

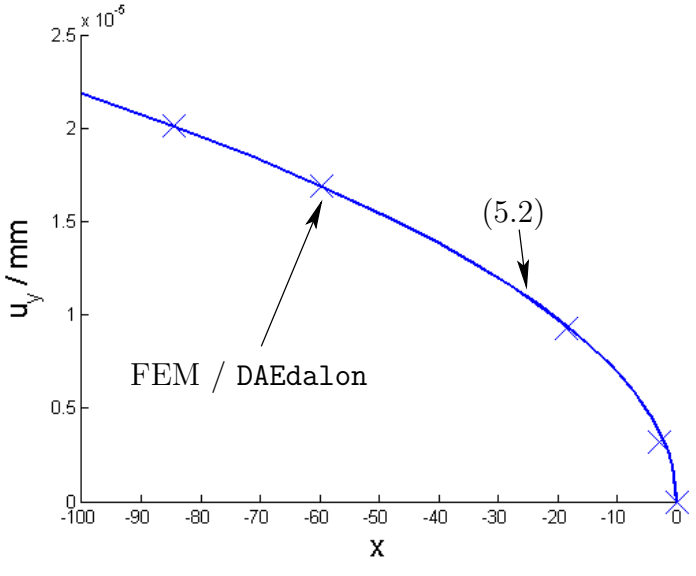

Fig. 5.7 Comparison of Analytical Solution (*line*) and FEM Results (×) for (5.2)

5.3.2 Analysis of Plastic Zone within K_I–Field

Furthermore, a straight forward extension of the above linear elastic treatment is the application of a material model incorporating plastic behavior in order to analyze the *plastic zone* near the crack tip ("*small scale yielding*"). Exemplarily, one may use simple VON MISES plasticity with a hardening rule as given in (3.26) and implemented in Sec. 4.6.5 of the form

$$\sigma_{yield} = \sigma_{yield\,0} \left[\frac{\alpha}{\varepsilon_0} + 1 \right]^{\frac{1}{N}} \tag{5.3}$$

with $\varepsilon_0 = \sigma_{yield\,0}/E$ and $\sigma_{yield\,0}$, E and N as material parameters.

For this example we use the elastic parameters as before and $\sigma_{yield\,0} = 210$ MPa as initial yield stress and $N = 6$ as hardening coefficient, respectively. Again, the example input files can be obtained from `www.DAEdalon.org/Examples/KI-field/plastic` to follow the descriptions here. The applied loading is now given by the loading factor $K_I = 100$ MPa$\sqrt{\text{m}}$, which we increase by

```
loop(140)
defo_scal=10
cont(13)
```

up to 140 times and observe the result by the (accumulated) equivalent plastic strain α. The resulting contour plot is given in a zoom–out in front of the crack tip in Fig. 5.8.

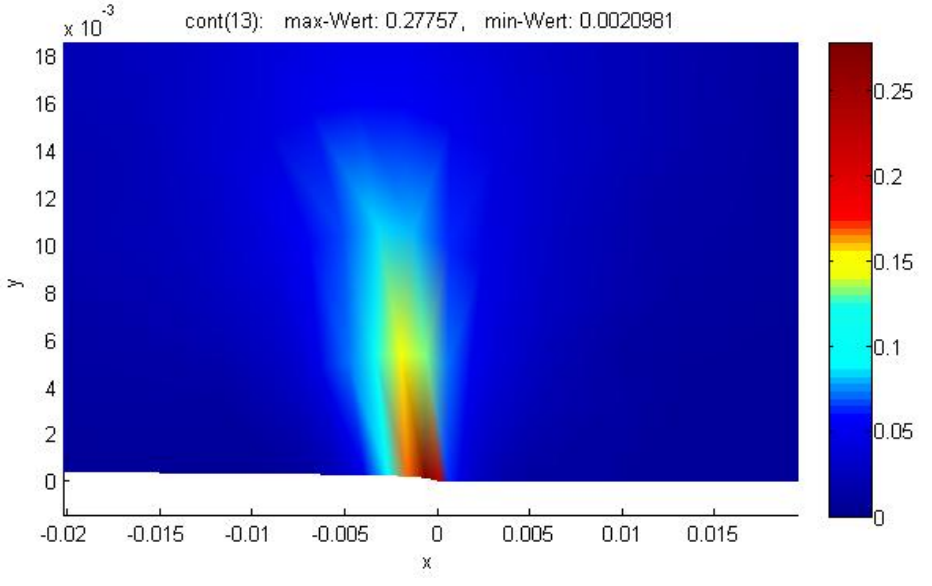

Fig. 5.8 Plastic zone indicated by the equivalent plastic strain α

6

What Does DAEdalon Mean ? — Background for Computational System — and Greek Mythology

- DAEdalon —
  Some years ago, in 2000, as we started with the basic concept and architecture of DAEdalon, along with the main, first focus of the FE system — namely the numerical treatment in the sense of a global system of *Differential Algebraic Equations*, see Eckert *et al.* [2003] and Ellsiepen & Hartmann [2001] — we intend to find a senseful name, which was also available to be used as web domain. Because, already at the beginning, it was clear to *open* the *source* of that code; on the one hand to give an easy as possible contact to the topic and to the basics of the FEM, which are published in many very good articles and other books; on the other hand to get the possibility of a broad community using and optimize the code within a diploma thesis, a PhD thesis or just in daily use. So, the artificial name DAEdalon seemed to fulfill our ideas and conditions nicely. At the same time, that name remembers quite well on the famous Greek "engineer" and his ideas.

- Daedalus & Ikarus —
  A partly reformulated text from Wikipedia*, the free encyclopedia*
  Daedalus was a most skillful artificer and was even said to have first invented images. He is first mentioned in Homer, where he built for Ariadne a wide dancing-ground (Iliad xviii.591). Homer still calls her by her Cretan title, the "Lady of the Labyrinth" (Iliad xviii.96) which Daedalus also made, in which the Minotaur was kept and from which Theseus escaped by means of the thread clue of Ariadne. Ignoring Homer, later writers envisaged the labyrinth as an edifice, and rather than a single path to the center and out again, gave it numberless winding passages and turnings opening into one another, seeming to have neither beginning nor end. Daedalus built it for King Minos, who needed the labyrinth to imprison his wife's son: Asterius, the Minotaur. Daedalus had built for Minos' wife, Pasiphae, a wooden cow so she could mate with the bull, for the Greeks imagined the Minoan bull of the sun to be an actual, earthly bull. Athenian pride made of Daedalus the grandson of the ancient king Erechtheus, who fled to Crete, having killed his nephew. Following other anecdotes,

Daedalus was shut up in a tower to prevent his knowledge of the labyrinth from spreading to the public. He could not leave Crete by sea, as the king kept strict watch on all the vessels, and permitted none to sail without being carefully searched. Since Minos controlled the land and sea routes, Daedalus set to work to fabricate wings for himself and his young son Icarus. He tied feathers together beginning with the smallest and adding larger, so as to form an increasing surface. The larger ones he secured with thread and the smaller with wax, and gave the whole a gentle curvature like the wings of a bird. When at last the work was done, the artist, waving his wings, found himself buoyed upward and hung suspended, poising himself on the beaten air. He next equipped his son in the same manner, and taught him how to fly. When both were prepared for flight, Daedalus warned Icarus not to fly too high, because the heat of the sun would melt the wax, nor too low because the sea foam would make the wings wet and they would no longer fly. Then the father and son flew away. They had passed Samos, Delos and Lebynthos when the boy began to soar upward as if to reach heaven. The blazing sun softened the wax which held the feathers together, and they came off. Icarus fell into the sea. His father cried and bitterly lamenting his own arts, called the land near the place where Icarus fell into the ocean Icaria in memory of his child. Daedalus arrived safe in Sicily in the care of King Cocalus, where he built a temple to Apollo, and hung up his wings, an offering to the god. Minos, meanwhile, searched for Daedalus by traveling from city to city asking a riddle. He presented a spiral seashell and asked for it to be strung all the way through. When he reached Camicus, King Cocalus, knowing Daedalus would be able to solve the riddle, fetched the old man. He tied the string to an ant, which walked through the seashell, stringing it all the way through. Minos then knew Daedalus was in the court of King Cocalus and demanded he be handed over. Cocalus managed to convince Minos to take a bath first. Cocalus' daughters then killed Minos. Daedalus was so proud of his achievements that he could not bear the idea of a rival. His sister had placed her son Perdix under his charge to be taught the mechanical arts. He was an apt scholar and gave striking evidences of ingenuity. Walking on the seashore he picked up the spine of a fish. Imitating it, he took a piece of iron and notched it on the edge, and thus invented the saw. He put two pieces of iron together, connecting them at one end with a rivet, and sharpening the other ends, and made a pair of compasses. Daedalus was so envious of his nephew's performances that he took an opportunity, when they were together one day on the top of a high tower, to push him off. But Athena, who favors ingenuity, saw him falling, and arrested his fate by changing him into a bird called after his name, the partridge. This bird does not build his nest in the trees, nor take lofty flights, but nestles in the hedges, and mindful of his fall, avoids high places. For this crime, Daedalus was tried and banished.

References

Andelfinger, U., Ramm, E.: Eas–elements for two–dimensional, three–dimensional, plate and shell structures and their equivalence to hr–elements. Int. J. Num. Meth. Eng. 36, 1311–1337 (1993)

Aravas, N.: On the numerical integration of a class of pressure–dependent plasticity models. Int. J. Num. Meth. Eng. 24, 1395–1416 (1987)

Baaser, H., Tvergaard, V.: A new algorithmic approach treating nonlocal effects at finite rate-independent deformation using the Rousselier damage model. Computer Methods in Applied Mechanics and Engineering 192(1-2), 107–124 (2003)

Belytschko, T., Ong, J., Liu, W.K., Kennedy, J.M.: Hourglass control in linear and nonlinear problems. Comp. Meth. Appl. Mech. Eng. 43, 251–276 (1984)

Dhatt, G., Touzot, G.: The Finite Element Method Displayed. Wiley, Chichester (1985)

Eckert, S., Baaser, H., Gross, D.: A DAE–approach applied to elastic–plastic material behavior using FEM. PAMM 3(1), 270–271 (2003)

Ellsiepen, P., Hartmann, S.: Remarks on the interpretation of current non–linear finite element analyses as differential–algebraic equations. Int. J. Num. Meth. Eng. 51, 679–707 (2001)

Gross, D., Seelig, T.: Fracture Mechanics. With an Introduction to Micromechanics. Springer, Heidelberg (2006)

Holzapfel, G.A.: Nonlinear Solid Mechanics. Wiley, Chichester (2000)

Hughes, T.J.R.: The Finite Element Method: Linear Static and Dynamic Finite Element Analysis. Dover Publications, Mineola (2000)

Miehe, C.: Aspects of the formulation and finite element implementation of large strain isotropic elasticity. Int. J. Num. Meth. Eng. 37, 1981–2004 (1994)

Reese, S., Wriggers, P.: A material model for rubber–like polymers exhibiting plastic deformation: computational aspects and a comparison with experimental results. Comp. Meth. Appl. Mech. Eng. 148, 279–298 (1997)

Simo, J.C.: Algorithms for static and dynamic multiplicative plasticity that preserve the classical return mapping schemes of the infinitesimal theory. Comp. Meth. Appl. Mech. Eng. 99, 61–112 (1992)

Simo, J.C., Armero, F.: Geometrically nonlinear enhanced–strain mixed methods and the method of incompatible modes. Int. J. Num. Meth. Eng. 33, 1413–1449 (1992)

Simo, J.C., Hughes, T.J.R.: Computational Inelasticity. Springer, New York (1998)

Stein, E., DeBorst, R., Hughes, T.J.: Encyclopedia of Computational Mechanics. Wiley, Chichester (2004)

Tsakmakis, C., Willuweit, A.: Use of the elastic predictor–plastic corrector method for integrating finite deformation plasticity laws. In: Hutter, K., Baaser, H. (eds.) Deformation and Failure in Metallic Materials. Springer, Heidelberg (2003)

Weber, G., Anand, L.: Finite deformation constitutive equations and a time integration procedure for isotropic, hyperelastic–viscoplastic solids. Comp. Meth. Appl. Mech. Eng. 79, 173–202 (1990)

Wriggers, P.: Nichtlineare Finite–Element–Methoden. Springer, Heidelberg (2001)

Index