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

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

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

$$c = \mathbf{a}^T \cdot \mathbf{b} = \{a_i b_i\} \quad (2.2)$$

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

Further on, we symbolize the dyadic product by

$$\mathbf{T} = \{T_{ij}\} = \mathbf{a} \otimes \mathbf{b} = \{a_i b_j\} \quad (2.3)$$

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

$$\text{trace}(\mathbf{T}) = \text{tr}(\mathbf{T}) = T_{11} + T_{22} + T_{33} = \{T_{ii}\}$$

$$= \{T_{ij} \delta_{ij}\} = \mathbf{T} : \mathbf{I}, \quad (2.4)$$
which can also be seen as a double contraction of that tensor by the identity I. For the index notation, here we introduce the Kronecker symbol $\delta_{ij}$ for 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 $T$, one may result in

$$
T^* = Q \cdot T \cdot Q^T
$$

with $Q^T = Q^{-1}$, det $Q = 1$ and symmetric $T = T^T$, where $T^*$ is a diagonalized array. The coefficients $T_1^*, T_2^*, T_3^*$ of $T^*$ are the eigenvalues of $T$, whereas $Q$ is composed by the eigenvectors $n_1, n_2$ and $n_3$ in the sense $Q = [n_1 n_2 n_3]$.

Equivalently, we give the spectral decomposition

$$
T = \sum_{k=1}^{3} T_k^* n_k \otimes n_k
$$

of the tensor $T$, 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 $e_1, e_2$ and $e_3$, which is justified in our sense by a consequent treatment in $\mathbb{R}^3$, 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 $x(X, t)$ and the corresponding displacement vector $u(X, t)$

$$
x = x(X, t) = X + u(X, t)
$$

with $X$ 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

$$
v = \dot{x} = \frac{\partial x}{\partial t}
$$

as material time derivative of the spatial position $x$.

Further on, the spatial gradient

$$
\text{grad } v = \frac{\partial v}{\partial x} =: 1
$$

of that field plays a crucial role in Sec. 3.2.3.
2.2 Kinematics and Deformation Gradient $F$

Finally, the deformation gradient is given by

$$F(X, t) = \text{Grad} x(X, t) = I + \text{Grad} u(X, t), \quad (2.10)$$

where the $\text{Grad}()$ operator is defined as $\frac{\partial(\cdot)}{\partial X}$ 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 $B_0$ in capital letters, while such of the actual configuration $B$ 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

$$F = R \cdot U = v \cdot R, \quad (2.11)$$
where $U$ and $v$ are known as right stretch tensor and left stretch tensor, respectively, and $R$ denotes a rotational tensor with $\det R = 1$ and $R^T R = I$. Exemplary, here we give a very efficient realization of a computation of the polar decomposition based on strain tensors in (2.17) resulting $U$, $R$ and $\lambda_1, \lambda_2, \lambda_3$ known as principal stretches, see Sec. 2.3.2.

```matlab
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
\[ J := \det F = \frac{dv}{dV} \]
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. 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 $F$ is given by
\[ \dot F = \text{Grad} \mathbf{v} = \mathbf{l} \cdot F, \]
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 $\mathbf{l}$

\[ ^1 \text{Please do not mix up the tensor } v \text{ in (2.11) with the velocity } \mathbf{v} \text{ some lines above or our symbol } v \text{ for the actual volume under consideration.} \]
2.3 Strain– and Stress–Tensors

\[ d := \frac{1}{2}(I + I^T) \]  \hspace{1cm} (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 \( N_I, I = 1, \ldots, nel. \)

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

\[ F = I + \sum_{I=1}^{nel} \frac{\partial N_I}{\partial \mathbf{x}} \cdot \mathbf{u}_I, \]  \hspace{1cm} (2.15)

which is given in DAEdalon in just one line as

\[ F = \text{eye}(3) + \text{u}_\text{elem'} \ast \text{dshape}; , \]

whenever the derivatives of \( N_I \) are given ordered in \text{dshape}.

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

Analogously, we are able to compute \( F \) in terms given in the actual configuration \( B \) by

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

as e.g. given in \text{defgrad_x.m}.

2.3 Strain– and Stress–Tensors

2.3.1 A Selection

We restrict ourselves to two representing strain tensors, named \text{GREEN–LAGRANGE} tensor \( \mathbf{E} \) and \text{ALMANSI} tensor \( \mathbf{e} \), acting on the reference configuration \( B_0 \) and the current configuration \( B \), respectively, see Fig. 2.1.

These are defined by

\[ \mathbf{E} := \frac{1}{2}(\mathbf{C} - \mathbf{I}) \]  \hspace{1cm} (2.17)

with the \text{right CAUCHY–GREEN} tensor \( \mathbf{C} := \mathbf{F}^\text{T} \cdot \mathbf{F} \) and
with the left Cauchy–Green tensor 
\[ \mathbf{b} := \mathbf{F} \cdot \mathbf{F}^T \]
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

\[
I_1(\mathbf{C}) = I_1(\mathbf{b}) = \text{trace} \mathbf{C} = \text{trace} \mathbf{b} = \lambda_1^2 + \lambda_2^2 + \lambda_3^2 ,
\]

while the second and third invariant write as

\[
I_2(\mathbf{C}) = I_2(\mathbf{b}) = \frac{1}{2} \left[ (\text{trace} \mathbf{C})^2 + \text{trace} \mathbf{C}^2 \right] = \text{trace} \mathbf{C}^{-1} \det \mathbf{C} = \lambda_1^2 \lambda_2^2 + \lambda_1 \lambda_2 \lambda_3^2 + \lambda_1 \lambda_2 \lambda_3^2 ,
\]

and

\[
I_3(\mathbf{C}) = I_3(\mathbf{b}) = \det \mathbf{C} = J^2 = \lambda_1^2 \lambda_2 \lambda_3^2 .
\]

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

\[
\frac{\partial I_1}{\partial \mathbf{C}} = \frac{\partial \text{trace} \mathbf{C}}{\partial \mathbf{C}} = \frac{\partial \mathbf{I} : \mathbf{C}}{\partial \mathbf{C}} = \mathbf{I} ,
\]

\[
\frac{\partial I_2}{\partial \mathbf{C}} = \frac{1}{2} (2 \text{trace} \mathbf{C} \mathbf{I} - \frac{\partial \text{trace} \mathbf{C}^2}{\partial \mathbf{C}}) = I_1 \mathbf{I} - \mathbf{C}
\]

and

\[
\frac{\partial I_3}{\partial \mathbf{C}} = I_3 \mathbf{C}^{-1} .
\]

### 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 \( \mathbf{F}_{\text{int}} \) 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

\[
\mathbf{t} = \lim_{\Delta a \to 0} \frac{\mathbf{F}_{\text{int}}}{\Delta a}
\]
in the limit of the considered area partition \( \Delta a \). 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

\[
\mathbf{t} = \mathbf{\sigma} \cdot \mathbf{n}_{\Delta a},
\]

the stress vector \( \mathbf{t} \) is also represented by the second order stress tensor \( \mathbf{\sigma} \) 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 \( \mathbf{\sigma} \), 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

\[
\mathbf{\sigma} = \begin{bmatrix}
\sigma_{11} \\
\sigma_{22} \\
\sigma_{33} \\
\sigma_{12} \\
\sigma_{23} \\
\sigma_{13}
\end{bmatrix}.
\]

**Remark**

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

\[
\mathbf{\varepsilon} = \begin{bmatrix}
\varepsilon_{11} \\
\varepsilon_{22} \\
\varepsilon_{33} \\
2 \varepsilon_{12} \\
2 \varepsilon_{23} \\
2 \varepsilon_{13}
\end{bmatrix}.
\]
For that reason, the ordering scheme of the coefficients of a *forth order material tensor* $\mathbf{D}$, 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

$$
\pi_{\text{int}} = \frac{1}{V} P_{\text{int}} = \mathbf{\sigma} : \mathbf{d}
$$  \hspace{1cm} (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

$$
\pi_{\text{int}}^0 = \frac{1}{V} P_{\text{int}} = \mathbf{S} : \dot{\mathbf{E}}
$$  \hspace{1cm} (2.31)

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

$$
\mathbf{S} := J \mathbf{F}^{-1} \cdot \mathbf{\sigma} \cdot \mathbf{F}^{-T}
$$  \hspace{1cm} (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 $\pi_{\text{int}}$ 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

$$
g(u, \delta u) := \int_{\mathcal{B}} \mathbf{\sigma} : \text{grad} \delta \mathbf{u} \, dv - \int_{\partial \mathcal{B}_o} \mathbf{t}_L \cdot \delta \mathbf{u} \, da = 0 ,
$$  \hspace{1cm} (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 $\mathbf{\sigma}$ the CAUCHY stress tensor is characterized and $\mathbf{t}_L = \mathbf{\sigma} \cdot \mathbf{n}$
are the prescribed tractions acting on the loaded boundary $\partial B_x$ of the body with (outer) normal vector $n$ in the current configuration $B$. 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 are many more types and enhancement on such ‘simple’ representations like (2.33); but for our basic explanations and discussion 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 $\sigma$ is given as outlined in (1.2) in vector form in a sense following the VOIGT notation $\bar{\sigma}$. In doing so, we are able to give

$$g(u, \delta u) = \int_B \text{grad} \delta u \cdot \bar{\sigma} \, dv - \int_{\partial B} \delta u \cdot t_L \, da = 0, \quad (2.34)$$

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

$$g(u, \delta u) \approx A g^{elmt}(u, \delta u). \quad (2.35)$$

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

$$g^{elmt}(u, \delta u) = \int_B \text{grad} \delta u^e \cdot \bar{\sigma}^e \, dv^{elmt} \quad (2.36)$$

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 serendipity– 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 \( \text{nél} - 1 \) nodes of the element the value of the shape function has to be 0.**
- **The sum of all \( \text{nél} \) shape functions at any position \( \xi \) in an element has to be 1:**

\[
\sum_{i=1}^{\text{nél}} N_i(\xi) = 1.
\]

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 \( l_e \):

\[
N_1(\xi) = 1 - \frac{\xi}{l_e} \quad \text{and} \quad N_2(\xi) = \frac{\xi}{l_e},
\]

\[\text{Eq. (2.37)}\]

![Fig. 2.3 Linear Shape Functions for 2–Noded Element of Length \( l_e \)](image-url)
2.6 Discretization in Space

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

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

\[ N_1(\xi, \eta) = \frac{1}{4} (1 - \xi)(1 - \eta) \]

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 \(B\) is described by

\[ u^e = \sum_{I=1}^{\text{nel}} N_I(\xi) u_I , \]

where \(I\) counts over all \(\text{nel}\) element nodes and \(u_I\) represents the discrete nodal displacement vector.

2.6.3 Derivatives of the Shape Functions

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

\[ \frac{\partial N_I}{\partial x} = J^{-T} \frac{\partial N_I}{\partial \xi} , \]

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

\[ \frac{\partial N_1(\xi, \eta)}{\partial \xi} = -\frac{1}{4} (1 - \eta) . \]
Applying the chain rule
\[
\frac{\partial N_I}{\partial \xi} = \frac{\partial N_I}{\partial x} \frac{\partial x}{\partial \xi} + \frac{\partial N_I}{\partial y} \frac{\partial y}{\partial \xi},
\]
\[
\frac{\partial N_I}{\partial \eta} = \frac{\partial N_I}{\partial x} \frac{\partial x}{\partial \eta} + \frac{\partial N_I}{\partial y} \frac{\partial y}{\partial \eta},
\]
we obtain \( J \) as
\[
J = \begin{bmatrix}
\frac{\partial x}{\partial \xi} & \frac{\partial x}{\partial \eta} \\
\frac{\partial y}{\partial \xi} & \frac{\partial y}{\partial \eta}
\end{bmatrix}.
\]
Here, the coefficients of \( J \) are computed by e.g.
\[
\frac{\partial x}{\partial \xi} = \sum_{I=1}^{nel} \frac{\partial N_I(\xi,\eta)}{\partial \xi} x_I
\]
following the isoparametric concept, which enables us in the end to obtain the derivatives of the shape functions with respect to the physical coordinates \( x \) in (2.40).

**Remark**
As \( J \) in (2.12) gives the volume ratio \( \frac{dV}{dV_0} \), here \( J \) 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 DAEEdalon 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 \( \frac{\partial N_I}{\partial x} \). The structure of that function follows the description 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);
```
2.6 Discretization in Space

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

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

% transformation
dsrpe=nsrpe*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

\[ I = \int_{-1}^{+1} g(\xi) \, d\xi \] (2.45)

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

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

where we can choose \( n \) weights \( w_j \) and \( n \) supporting points \( \xi_j \) 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 $\xi_1, \xi_2$ in the case of $n = 2$ in (2.46):

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

Comparing coefficients in (2.47) gives

$$2k_0 = (w_1 + w_2) \, k_0$$
$$\frac{2}{3} k_2 = (w_1 \xi_1^2 + w_2 \xi_2^2) \, k_2$$
$$0 = (w_1 \xi_1 + w_2 \xi_2) \, k_1$$
$$0 = (w_1 \xi_1^3 + w_2 \xi_2^3) \, k_3$$  \hspace{1cm} (2.48)

resulting in four equations (2.48)\textsubscript{1} to (2.48)\textsubscript{4} for the four unknowns $w_1, w_2, \xi_1$ and $\xi_2$. In order to result in a symmetric distribution of the both supporting points $\xi_1$ and $\xi_2$, see Fig. 2.5, with equal weighting factors $w_1, w_2$, so that

$$w : = w_1 = w_2$$
$$\xi : = \xi_1 = -\xi_2 ,$$  \hspace{1cm} (2.49)

we obtain directly

$$w = w_1 = w_2 = 1$$
$$\xi = \xi_1 = -\xi_2 = \frac{1}{\sqrt{3}} \approx 0.57735$$  \hspace{1cm} (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 choosing 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 polynomials 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

```matlab
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–nodded prototype element with a \(2 \times 2\) integration scheme in \(\xi\)– and \(\eta\)–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

\[
g^{\mathrm{elm}}(u, \delta u) \approx \delta u^e \cdot \sum_n B_n^e \cdot \bar{\sigma}_n \cdot w_n
\]

results, where in the \(B\)–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 \(u\) 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

\[
\Delta g^{\text{elm}}(\hat{\mathbf{u}}, \delta \mathbf{u}) = \int_{G^{\text{elm}}} \left( \Delta \mathbf{\sigma} + \text{grad} \Delta \mathbf{u} \cdot \mathbf{\sigma} \right) : \text{grad} \delta \mathbf{u} \, d\mathbf{v}^{\text{elm}}, \tag{2.53}
\]

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 \( \mathbf{K}^{\text{elm}} \) for the discretized setting, where \( \mathbf{K}^{\text{elm}} \) obviously consists of two parts. The first part \( \mathbf{K}_{\text{mat}} \) is obtained from the consistent linearization of the material model getting \( \Delta \mathbf{\sigma} \), see Chap. 3, and the second part \( \mathbf{K}_{\text{geom}} = \int k_{\text{geom}} d\mathbf{v}^{\text{elm}} \) comes solely from the linearization of the used strain measure at the computed stress state \( \mathbf{\sigma} \).

### 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

\[
\delta \mathbf{u} \cdot \left\{ \Delta g(\hat{\mathbf{u}}) \cdot \Delta \mathbf{u} + g(\hat{\mathbf{u}}) \right\} = 0 \tag{2.54}
\]

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.