Exercises to Finite Element Methods

Within this exercise we develop our own simple Finite Elements Code. Doing so, we will restrict ourselves to two-dimensional (shell-) problems and linear elasticity with small displacements. Those relatively strict limitations allow us, by having a still managable and surveyable code, to focus on the substantial aspects of the finite elements method.

- Formulation of elements, geometrical mapping, Interpolation functions,
- Description of material, stiffness matrix, numeric integration,
- Boundary conditions, Load vector,
- Assembly and
- Application of the FE-Program as well as postprocessing.

1 Interpolation functions / shape functions

Exercise 1.1 Interpolation functions - the basis of each element

For a bilinear rectangular element the interpolation functions are given in natural/parent coordinates ξ and η as follows:

$$\begin{aligned} N_1 &= \frac{1}{4}(1+\xi)(1+\eta), \\ N_2 &= \frac{1}{4}(1-\xi)(1+\eta), \\ N_3 &= \frac{1}{4}(1-\xi)(1-\eta), \\ N_4 &= \frac{1}{4}(1+\xi)(1-\eta), \end{aligned} \tag{1.1}$$

see figure 1.1.

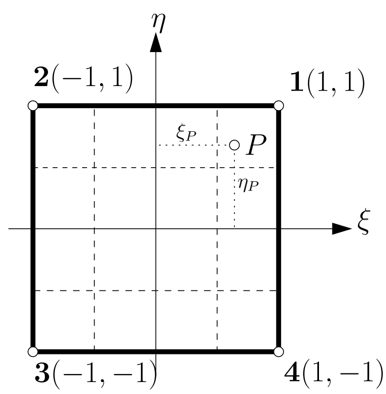

Figure 1.1: Natural coordinates ξ and η of a rectangular element.

The interpolation functions may in principle be used for the interpolation of arbitrary field quantities. We begin with an interpolation of a simple scalar function f:

$$f(\xi,\eta) = \sum_{i=1}^{4} N_i(\xi,\eta) f_i = N_1(\xi,\eta)f_1 + N_2(\xi,\eta)f_2 + N_3(\xi,\eta)f_3 + N_4(\xi,\eta)f_4 \tag{1.2}$$

with the nodal values f_i at the corresponding nodes i, see figure 1.2.

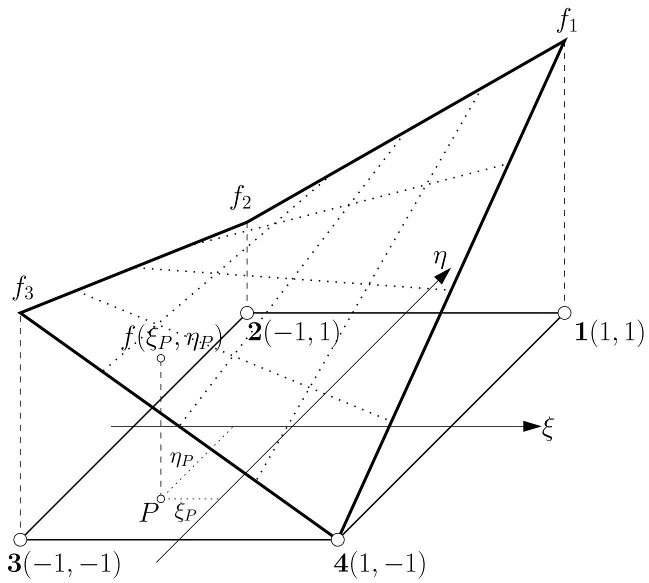

Figure 1.2: Darstellung der Interpolationsfunktion $f = f(\xi, \eta)$

Exercises:

- Write a function (shapefunc.m) for the computation of the interpolation functions N_i at coordinates ξ, η (d.h. Input: ξ, η; Output: $N_i = [N1, N2, N3, N4]$).

```
function[Ni] = shapefunc(xsi,eta)
    ...
end
```

- Write a script-file for the computation of the function value f at coordinates ξ_P, η_P with the nodal values f_i

  (Test for $f_1 = 5,\ f_2 = 1,\ f_3 = 3,\ f_4 = 0$; $\xi_P = 1/\sqrt{3}$, $\eta_P = -1/\sqrt{3}$; Result: $f_P = 1.378$).

- Write a script-files for the graphical illustration of function $f(\xi, \eta)$.

Exercise 1.2 Use of interpolation functions for geometrical mapping

With the interpolation functions N_i from equ. (1.1) one may describe the geometrical mapping shown in fig. 1.3 as follows:

$$\begin{aligned} x(\xi,\eta) &= \sum_{i=1}^{4} N_i(\xi,\eta)x_i &&= N_1(\xi,\eta)x_1 + N_2(\xi,\eta)x_2 + N_3(\xi,\eta)x_3 + N_4(\xi,\eta)x_4, \\ y(\xi,\eta) &= \sum_{i=1}^{4} N_i(\xi,\eta)y_i &&= N_1(\xi,\eta)y_1 + N_2(\xi,\eta)y_2 + N_3(\xi,\eta)y_3 + N_4(\xi,\eta)y_4, \end{aligned} \tag{1.3}$$

with x_i as the nodal value of coordinate x at node i, and y_i as the nodal value of coordinate y at node i.

Equation (1.3) describes an interpolated vector function $\mathbf{x}(\xi,\eta)$ and may be written in terms of a vector as follows:

$$\mathbf{x}(\xi,\eta)=\sum_{i=1}^{4}N_i(\xi,\eta)\mathbf{x}_i=N_1(\xi,\eta)\mathbf{x}_1+N_2(\xi,\eta)\mathbf{x}_2+N_3(\xi,\eta)\mathbf{x}_3+N_4(\xi,\eta)\mathbf{x}_4 \tag{1.4}$$

with $\mathbf{x}_i=\begin{pmatrix}x_i\\ y_i\end{pmatrix}\ldots$ position vector of node i.

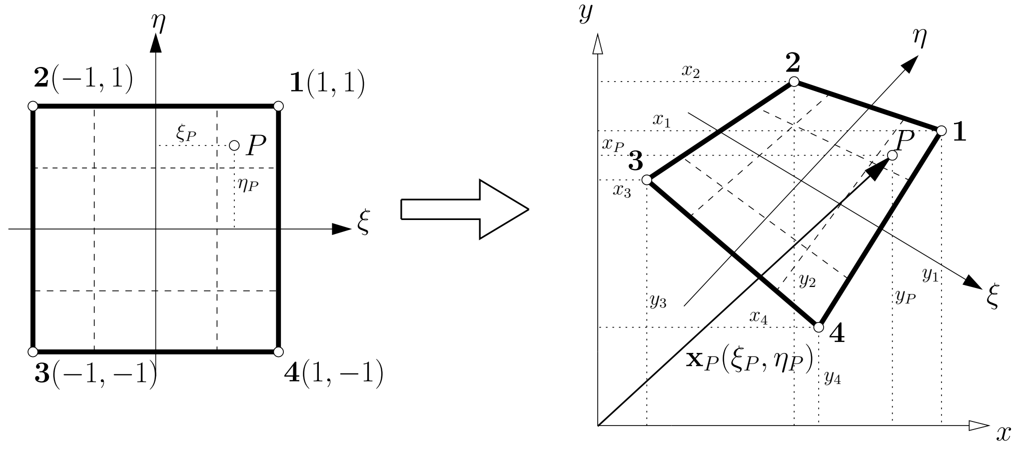

Figure 1.3: Geometrical mapping $\mathbf{x}=\mathbf{x}(\xi,\eta)$

Exercise:

- Write a `script`-file for the computation of the function value $\mathbf{x}$ at coordinates ξ,η with the given nodes $\mathbf{x}_i$, as well as for a graphical illustration of the geometrically mapped square and node P - given through the position vector $\mathbf{x}_P$ - see fig. 1.3.
  (Test for coordinates $\mathbf{x}_1=(5.5,5)$, $\mathbf{x}_2=(3,6)$, $\mathbf{x}_3=(1,4)$, $\mathbf{x}_4=(3.5,1.5)$ as well as for $\xi_P=1$, $\eta_P=0.5$)

Exercise 1.3 Partial Derivations of the interpolation functions

The partial derivations of the interpolation functions with respect to ξ and η are:

$$\begin{aligned}
\frac{\partial N_1}{\partial\xi} &= \tfrac{1}{4}(1+\eta) & \frac{\partial N_1}{\partial\eta} &= \tfrac{1}{4}(1+\xi)\\
\frac{\partial N_2}{\partial\xi} &= -\tfrac{1}{4}(1+\eta) & \frac{\partial N_2}{\partial\eta} &= \tfrac{1}{4}(1-\xi)\\
\frac{\partial N_3}{\partial\xi} &= -\tfrac{1}{4}(1-\eta) & \frac{\partial N_3}{\partial\eta} &= -\tfrac{1}{4}(1-\xi)\\
\frac{\partial N_4}{\partial\xi} &= \tfrac{1}{4}(1-\eta) & \frac{\partial N_4}{\partial\eta} &= -\tfrac{1}{4}(1+\xi)
\end{aligned} \tag{1.5}$$

The interpolation of these scalar functions $\frac{\partial f}{\partial\xi}$ und $\frac{\partial f}{\partial\eta}$ therefore is given to

$$\begin{aligned}
\frac{\partial f(\xi,\eta)}{\partial\xi} &= \sum_{i=1}^{4}\frac{\partial N_i(\xi,\eta)}{\partial\xi}f_i=\frac{\partial N_1(\xi,\eta)}{\partial\xi}f_1+\frac{\partial N_2(\xi,\eta)}{\partial\xi}f_2+\frac{\partial N_3(\xi,\eta)}{\partial\xi}f_3+\frac{\partial N_4(\xi,\eta)}{\partial\xi}f_4\\
\frac{\partial f(\xi,\eta)}{\partial\eta} &= \sum_{i=1}^{4}\frac{\partial N_i(\xi,\eta)}{\partial\eta}f_i=\frac{\partial N_1(\xi,\eta)}{\partial\eta}f_1+\frac{\partial N_2(\xi,\eta)}{\partial\eta}f_2+\frac{\partial N_3(\xi,\eta)}{\partial\eta}f_3+\frac{\partial N_4(\xi,\eta)}{\partial\eta}f_4
\end{aligned} \tag{1.6}$$

$f_i\ldots$ nodal value at node i

Exercises:

- Write a `function` (`dshapefunc.m`) for the computation of the partial derivations of N_i ($\partial N_i/\partial\xi$ und $\partial N_i/\partial\eta$) at coordinates ξ, η on behalf of equ. (1.5).

  (Input: ξ, η; Output: $\partial N_i(\xi,\eta)/\partial\xi$ and $\partial N_i(\xi,\eta)/\partial\eta$, whereas i=1,2,3,4.)

```
function[dNidxsi,dNideta] = dshapefunc(xsi,eta)
    ...
end
```

- Write a `script`-file for the computation of $\partial f(\xi,\eta)/\partial\xi$ ans $\partial f(\xi,\eta)/\partial\eta$ at coordinates ξ_P, η_P with given nodal values f_i.
  (Test for $f_1 = 5$, $f_2 = 1$, $f_3 = 3$, $f_4 = 0$; $\xi_P = -0.5$, $\eta_P = -0.5$; Result: $\partial f(\xi,\eta)/\partial\xi = -0.625$, $\partial f(\xi,\eta)/\partial\eta = -0.125$)

- Adapt the `script`-file from Exercise 1.1 for the graphical illustration of the partial derivations f ($\partial f/\partial\xi$ und $\partial f/\partial\eta$).

Exercise 1.4 Derivation of the interpolation functions with respect to x and y with the help of the Jacobi-Matrix

Next, we want to compute the partial derivations of the interpolation functions N_i with respect to x and y with the help of the Jacobi-matrix.
Application of the chain rule results in:

$$\begin{bmatrix} \frac{\partial N_i}{\partial\xi} \\ \\ \frac{\partial N_i}{\partial\eta} \end{bmatrix} = \begin{bmatrix} \frac{\partial N_i}{\partial x}\frac{\partial x}{\partial\xi} & + & \frac{\partial N_i}{\partial y}\frac{\partial y}{\partial\xi} \\ \\ \frac{\partial N_i}{\partial x}\frac{\partial x}{\partial\eta} & + & \frac{\partial N_i}{\partial y}\frac{\partial y}{\partial\eta} \end{bmatrix} = \underbrace{\begin{bmatrix} \frac{\partial x}{\partial\xi} & \frac{\partial y}{\partial\xi} \\ \\ \frac{\partial x}{\partial\eta} & \frac{\partial y}{\partial\eta} \end{bmatrix}}_{\mathbf{J}} \begin{bmatrix} \frac{\partial N_i}{\partial x} \\ \\ \frac{\partial N_i}{\partial y} \end{bmatrix} i = 1,2,3,4 \tag{1.7}$$

with the Jacobi-Matrix $\mathbf{J}$. We derive

$$\begin{bmatrix} \frac{\partial N_i}{\partial x} \\ \\ \frac{\partial N_i}{\partial y} \end{bmatrix} = \underbrace{\begin{bmatrix} \frac{\partial\xi}{\partial x} & \frac{\partial\eta}{\partial x} \\ \\ \frac{\partial\xi}{\partial y} & \frac{\partial\eta}{\partial y} \end{bmatrix}}_{\mathbf{J}^{-1}} \begin{bmatrix} \frac{\partial N_i}{\partial\xi} \\ \\ \frac{\partial N_i}{\partial\eta} \end{bmatrix} \tag{1.8}$$

whereas (see Exercise 1.2)

$$\begin{aligned} J_{11} = \frac{\partial x(\xi,\eta)}{\partial\xi} &= \sum_{i=1}^{4} \frac{\partial N_i(\xi,\eta)}{\partial\xi} x_i = \frac{\partial N_1}{\partial\xi}x_1 + \frac{\partial N_2}{\partial\xi}x_2 + \frac{\partial N_3}{\partial\xi}x_3 + \frac{\partial N_4}{\partial\xi}x_4 \\ &= \begin{bmatrix} \frac{\partial N_1}{\partial\xi} & \frac{\partial N_2}{\partial\xi} & \frac{\partial N_3}{\partial\xi} & \frac{\partial N_4}{\partial\xi} \end{bmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} \\ J_{12} = \frac{\partial y(\xi,\eta)}{\partial\xi} &= \sum_{i=1}^{4} \frac{\partial N_i(\xi,\eta)}{\partial\xi} y_i = \frac{\partial N_1}{\partial\xi}y_1 + \frac{\partial N_2}{\partial\xi}y_2 + \frac{\partial N_3}{\partial\xi}y_3 + \frac{\partial N_4}{\partial\xi}y_4 \\ &= \begin{bmatrix} \frac{\partial N_1}{\partial\xi} & \frac{\partial N_2}{\partial\xi} & \frac{\partial N_3}{\partial\xi} & \frac{\partial N_4}{\partial\xi} \end{bmatrix} \cdot \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{pmatrix} \end{aligned} \tag{1.9}$$

and

$$
\begin{aligned}
J_{21} &= \frac{\partial x(\xi,\eta)}{\partial \eta} = \sum_{i=1}^{4} \frac{\partial N_i(\xi,\eta)}{\partial \eta} x_i = \frac{\partial N_1}{\partial \eta}x_1 + \frac{\partial N_2}{\partial \eta}x_2 + \frac{\partial N_3}{\partial \eta}x_3 + \frac{\partial N_4}{\partial \eta}x_4 \\
&= \begin{bmatrix} \frac{\partial N_1}{\partial \eta} & \frac{\partial N_2}{\partial \eta} & \frac{\partial N_3}{\partial \eta} & \frac{\partial N_4}{\partial \eta} \end{bmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} \\
J_{22} &= \frac{\partial x(\xi,\eta)}{\partial \eta} = \sum_{i=1}^{4} \frac{\partial N_i(\xi,\eta)}{\partial \eta} y_i = \frac{\partial N_1}{\partial \eta}y_1 + \frac{\partial N_2}{\partial \eta}y_2 + \frac{\partial N_3}{\partial \eta}y_3 + \frac{\partial N_4}{\partial \eta}y_4 \\
&= \begin{bmatrix} \frac{\partial N_1}{\partial \eta} & \frac{\partial N_2}{\partial \eta} & \frac{\partial N_3}{\partial \eta} & \frac{\partial N_4}{\partial \eta} \end{bmatrix} \cdot \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{pmatrix}
\end{aligned}
\tag{1.10}
$$

The values $\partial N_i/\partial \xi$ (dNdxsi) and $\partial N_i/\partial \eta$ (dNdeta) follow from `function dshapefunc.m`. Once we have written these derivations as well as the coordinates of the nodes in terms of row vectors, we may derive:

$$
\begin{aligned}
J_{11} &= \mathrm{dxdxsi} &= \mathrm{dNdxsi} \cdot \mathrm{x}' \\
J_{12} &= \mathrm{dydxsi} &= \mathrm{dNdxsi} \cdot \mathrm{y}' \\
J_{21} &= \mathrm{dxdeta} &= \mathrm{dNdeta} \cdot \mathrm{x}' \\
J_{22} &= \mathrm{dydeta} &= \mathrm{dNdeta} \cdot \mathrm{y}'
\end{aligned}
\tag{1.11}
$$

We then obtain the derivations $\partial N_i/\partial x$ (dNdx) and $\partial N_i/\partial y$ (dNdy) as follows

$$
\begin{bmatrix} \mathrm{dNdx} \\ \mathrm{dNdy} \end{bmatrix} = \mathrm{J}^{-1} \cdot \begin{bmatrix} \mathrm{dNdxsi} \\ \mathrm{dNdeta} \end{bmatrix}. \tag{1.12}
$$

Exercises:

- Write a `function` (`jacobi.m`) for the computation of the Jacobi-Matrix **J** at coordinates ξ, η of a square element with node coordinates (x_i, y_i).

  Input: $x = [x1, x2, x3, x4]$, $y = [y1, y2, y3, y4]$ and $\partial N_i(\xi,\eta)/\partial\xi$, $\partial N_i(\xi,\eta)/\partial\eta$, evaluated at coordinates ξ,η;

  Output: J

```
function[J] = jacobi(xi,yi,dNidxsi,dNideta)
    dxdxsi = dNdxsi*xi';
    ...
    J = [dxdxsi,dydxsi;dxdeta,dydeta];
end
```

- Write a `function` (`dxyshapefunc.m`) for the computation of the partial derivations of N_i with respect to x and y ($\partial N_i(\xi,\eta)/\partial x$ and $\partial N_i(\xi,\eta)/\partial y$) at coordinates ξ, η.

  Input: J, $\partial N_i(\xi,\eta)/\partial\xi$, $\partial N_i(\xi,\eta)/\partial\eta$, evaluated at coordinates ξ,η;

  Output: dNidx= $\partial N_i(\xi,\eta)/\partial x$ and dNidy= $\partial N_i(\xi,\eta)/\partial y$, evaluated at coordinates ξ,η

```
function [dNix,dNiy] = dxyshapefunc(J,dNdxsi,dNdeta)
    ...
end
```

- Write a `script`-file for the computation of the values f and its partial derivations f with respect to x and y $\big(\partial f(\xi,\eta)/\partial x$ and $\partial f(\xi,\eta)/\partial y\big)$ as well as for the graphical illustration of f, $\partial f(\xi,\eta)/\partial x$ and $\partial f(\xi,\eta)/\partial y$.
  (Test for node coordinates $\mathbf{x}_1 = (3,1)$, $\mathbf{x}_2 = (1,4)$, $\mathbf{x}_3 = (-1,4)$, $\mathbf{x}_4 = (1,-3)$ as well as for $f_1 = 0$, $f_2 = 1$, $f3 = 2$ and $f_4 = 1$)

Exercise 1.5 *Homework*

3-node-element with a linear displacement approach – CST-Element, see lecture notes for theory

Exercises:

- Write a `function` (`shapefunc_CST.m`) for the computation of the interpolation functions N_i at coordinates ξ_1, ξ_2

  (i.e. Input: ξ_1, ξ_2; Output: $N_i = [N1, N2, N3]$).

- Write a `function` (`dshapefunc_CST.m`) for the computation of the partial derivations of N_i ($\partial N_i/\partial \xi_1$ und $\partial N_i/\partial \xi_2$) at coordinates ξ_1, ξ_2

  (i.e. Input: ξ_1, ξ_2; Output: $\partial N_i(\xi_1, \xi_2)/\partial \xi_1$ and $\partial N_i(\xi_1, \xi_2)/\partial \xi_2$, whereas i=1,2,3.)

- Write a `script`-file (filename: `script_HUE1_1_Lastname.m`, please no special characters) for the computation of the function value f as well as the partial derivations $\partial f(\xi_1, \xi_2)/\partial \xi_1$ and $\partial f(\xi_1, \xi_2)/\partial \xi_2$ at coordinates $\xi_{1,P}, \xi_{2,P}$ with known nodal values f_i.
  Values for $f(0,0) = 1$, $f(1,0) = 0$, $f(0,1) = 1.5$, as well as $\xi_{1,P} = 0.5$, $\xi_{2,P} = 0.25$.

  Check:

  - $f(0.5, 0.25) = 0.625$
  - $\partial f/\partial \xi_1(0.5, 0.25) = -1$
  - $\partial f/\partial \xi_2(0.5, 0.25) = 0.5$

- Write a `script`-file (filename: `script_HUE1_2_Lastname.m`, please no special characters), which should reproduce the plot shown in figure 1.4, with use of the CST interpolation functions.

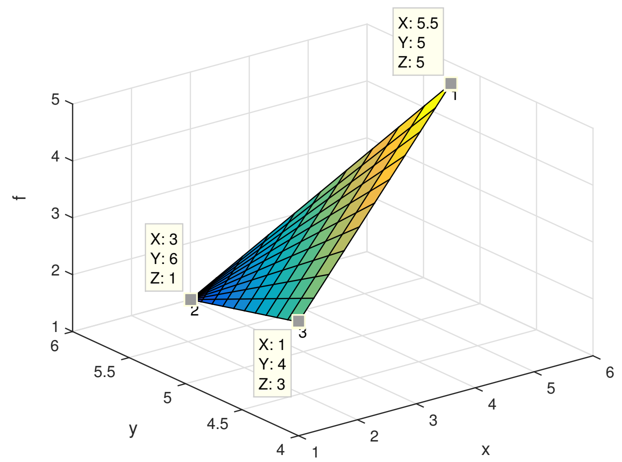

Figure 1.4: Geometrical mapping as well as interpolation with CST interpolation functions

Upload of files (as .zip-file) on TUWEL:

- the function `shapefunc_CST.m`
- the function `dshapefunc_CST.m`
- both `script`-files

Listings

- Exercise 1.1, Punkt 1:

Listing 1: code/Einheit1/EN/shapefunc.m

```
function[Ni] = shapefunc(xsi,eta)
%--------------------------------------------------------------------
%   Function for the computation of values Ni (i = 1,2,3,4)
%   at coordinates (xsi,eta)
%
%   Input:  xsi...within -1 <= xsi <= 1
%           eta...within -1 <= eta <= 1
%
%   Output: Ni....Ni = [N1,N2,N3,N4]; interpolation function
%
%   Syntax:
%   Ni = shapefunc(xsi,eta)
%
%   Copyright: IMWS
%   Version:   2 (EN)
%   Datum:     20151109
%--------------------------------------------------------------------

% Ensure xsi and eta are column vectors
xsi = xsi(:);
eta = eta(:);

%   Interpolation function Ni for bilinear 4-node element
%   at coordinates (xsi,eta)
N1  = 1/4*(1+xsi).*(1+eta);
N2  = 1/4*(1-xsi).*(1+eta);
N3  = 1/4*(1-xsi).*(1-eta);
N4  = 1/4*(1+xsi).*(1-eta);

%   Store values Ni in row vector
Ni  = [N1,N2,N3,N4];

end
```

- Exercise 1.1, Part 2:

Listing 2: code/Einheit1/EN/script_FEUE_1.m

```
%--------------------------------------------------------------------
%   Script for the computation of the values of the scalar
%   interpolation functions f with nodal values f1,f2,f3,f4 at
%   coordinates (xsip,etap)
%
%   Copyright: IMWS
%   Version:   2 (EN)
%   Datum:     20151109
%--------------------------------------------------------------------

clear
close all
clc

%% Enter nodal values
% f1.....value of function f at node 1; f1 = f(xsi = +1,eta = +1)
% f2.....value of function f at node 1; f2 = f(xsi = -1,eta = +1)
% f3.....value of function f at node 1; f3 = f(xsi = -1,eta = -1)
% f4.....value of function f at node 1; f4 = f(xsi = +1,eta = -1)
f1 = 5;
```

```
f2 = 1;
f3 = 3;
f4 = 0;

%% Computation
% Formation of a node row vector fi
fi = [f1,f2,f3,f4];

% Enter coordinates of node P
xsip = 1/sqrt(3);
etap = -1/sqrt(3);

% Computation of interpolation functions Ni(xsip,etap)
[Ni] = shapefunc(xsip,etap);

% Computation of function value at node P f(xsip,etap) = Ni(xsip,etap)*fi'
f  = Ni*fi(:)
```

- Exercise 1.1, Part 3:

Listing 3: code/Einheit1/EN/script_FEUE_2.m

```
%-------------------------------------------------------------------
%  Script for graphical illustration of the interpolation funktion
%  f(xsi,eta) with node values f1,f2,f3,f4
%
%  Copyright: IMWS
%  Version:   2 (EN)
%  Datum:     20151109
%-------------------------------------------------------------------

clear
close all
clc

%% Enter node values
% f1.....value of function f at node 1; f1 = f(xsi = +1,eta = +1)
% f2.....value of function f at node 1; f2 = f(xsi = -1,eta = +1)
% f3.....value of function f at node 1; f3 = f(xsi = -1,eta = -1)
% f4.....value of function f at node 1; f4 = f(xsi = +1,eta = -1)
f1 = 5;
f2 = 1;
f3 = 3;
f4 = 0;

%% Computation
% Formation of a node row vector fi
fi = [f1,f2,f3,f4];

% Enter value domain of xsi and eta as vectors
xsi = -1:0.1:1;
eta = -1:0.1:1;

lxsi = numel(xsi);
leta = numel(eta);

% Computation of function values within xsi = -1..+1,eta = -1..+1
% Initialize f
f = zeros(lxsi,leta);
xsiMat = zeros(lxsi,leta);
etaMat = zeros(lxsi,leta);

```

```
for i = 1:lxsi
   for j = 1:leta
      % Evaluate interpolation function at nodes xsi(i),
      % eta(j)
      Ni = shapefunc( xsi(i), eta(j) );
      f(i,j) = Ni*fi(:);

      % Save coordinates in additional Matrix
      xsiMat(i,j) = xsi(i);
      etaMat(i,j) = eta(j);
   end
end

%% Possibilities for the graphical illustration of interpolation functions

% Generate new figure
figure
% Generate surface object
surf(xsiMat,etaMat,f);
xlabel('\xi');
ylabel('\eta');
zlabel('f');
% surface objects may be evaluated interactively in 'datacursormode'

% Save figure
print -depsc abb_FEUE_2



% Generate new figure
figure
% Generate contour object
contour(xsiMat,etaMat,f)

% Generate new figure
figure
% Generate pseudo-color object
pcolor(xsiMat,etaMat,f)
```

- Exercise 1.2:

Listing 4: code/Einheit1/EN/script_FEUE_3.m

```
%-----------------------------------------------------------------------
%   Script for the computation of values of the interpolation vector
%   function X(xp,yp), i.e. the coordinates of node P at coordinates
%   (xsip,etap) within the x,y-system, with given node values
%   x1,y1,x2,y2,x3,y3,x4,y4
%
%   Copyright: IMWS
%   Version:   2
%   Datum:     20151109
%-----------------------------------------------------------------------

clear
close all
clc

%% Enter node values
% node 1 (x1,y1) -> (xsi = +1,eta = +1)
% node 2 (x2,y2) -> (xsi = -1,eta = +1)
% node 3 (x3,y3) -> (xsi = -1,eta = -1)
```

```
% node 4 (x4,y4) -> (xsi = +1,eta = -1)
x1 = 5.5;
y1 = 5;

x2 = 3;
y2 = 6;

x3 = 1;
y3 = 4;

x4 = 3.5;
y4 = 1.5;

%% computation
% Fit node values into vectors
x = [x1,x2,x3,x4];
y = [y1,y2,y3,y4];

% Enter coordinates of node P within the xsi,eta-system
xsip = 1;
etap = 0.5;

% Computation of the interpolation functions (xsip,etap)
[Ni] = shapefunc(xsip,etap);

% Computation of coordinates of node P within the x,y-system
xp = Ni*x(:);
yp = Ni*y(:);

%% Graphical illustration
% Generate new figure
figure

% Drawing of borders
plot([x x(1)],[y y(1)])

% Do not overwrite existing plot
hold on

% Draw node P
plot(xp,yp,'or')

% Assignment of node numbers
for knotenNo = 1:4
    text(x(knotenNo),y(knotenNo),num2str(knotenNo), ...
        'VerticalAlignment','bottom', ...
        'HorizontalAlignment','left');
end
```

- Exercise 1.3, Part 1:

Listing 5: code/Einheit1/EN/dshapefunc.m

```
function[dNidxsi,dNideta] = dshapefunc( xsi, eta )
%----------------------------------------------------------------------
%  Function for the computation of values dNi/dxsi resp. dNi/deta
%  (i=1,2,3,4) at coordinates (xsi,eta)
%
%  Input:
%          xsi ... for values -1 <= xsi <= 1
%          eta ... for values -1 <= eta <= 1
%
```

```
%  Output: dNidxsi ... dNi/dxsi =   [dN1/dxsi,dN2/dxsi,dN3/dxsi,dN4/dxsi]
%          dNideta ... dNi/deta = [dN1/deta,dN2/deta,dN3/deta,dN4/deta]
%
%  Syntax:
%  [dNixsi,dNieta] = dshapefunc( xsi, eta )
%
%  Assignment of node numbers:
%  node 1: xsi = +1,eta = +1
%  node 2: xsi = -1,eta = +1
%  node 3: xsi = -1,eta = -1
%  node 4: xsi = +1,eta = -1
%
%  Copyright: IMWS
%  Version:   2 (EN)
%  Datum:     20151109
%-------------------------------------------------------------------

%  Derivation of interpolation functions Ni for a bilinear 4-node element
%  at coordinates (xsi,eta) with respective to xsi
dN1dxsi = 1/4*(1+eta);
dN2dxsi = -1/4*(1+eta);
dN3dxsi = -1/4*(1-eta);
dN4dxsi = 1/4*(1-eta);

%  Formation of an interpolation row vector dNixsi
dNidxsi = [dN1dxsi,dN2dxsi,dN3dxsi,dN4dxsi];

%  Derivation of interpolation functions Ni for a bilinear 4-node element
%  at coordinates (xsi,eta) with respective to eta
dN1deta = 1/4*(1+xsi);
dN2deta = 1/4*(1-xsi);
dN3deta = -1/4*(1-xsi);
dN4deta = -1/4*(1+xsi);

%  Store values dNieta in row vector
dNideta = [dN1deta,dN2deta,dN3deta,dN4deta];

end
```

- Exercise 1.3, Part 2:

Listing 6: code/Einheit1/EN/script_FEUE_4.m

```
%-------------------------------------------------------------------
%  Script for the computation of the values of the scalar
%  interpolation functions dfxsi and dfeta with node values
%  f1,f2,f3,f4 at coordinates (xsip,etap)
%
%  Copyright: IMWS
%  Version:   2 (EN)
%  Datum:     20151109
%-------------------------------------------------------------------

clear
close all
clc

%% Enter node values
% f1.....value of f at node 1; f1 = f(xsi = +1,eta = +1)
% f2.....value of f at node 2; f2 = f(xsi = -1,eta = +1)
% f3.....value of f at node 3; f3 = f(xsi = -1,eta = -1)
% f4.....value of f at node 4; f4 = f(xsi = +1,eta = -1)
```

```
f1 = 5;
f2 = 1;
f3 = 3;
f4 = 0;

%% Computation
%  Formation of node row vector fi
fi = [f1,f2,f3,f4];

% Enter coordinates of node P
xsip = -0.5;
etap = -0.5;

%  Computation of functions dNixsi(xsip,etap) and dNieta(xsip,etap)
[dNidxsi,dNideta] = dshapefunc(xsip,etap);

% Computation of wanted function values dfxsi(xsip,etap)
% and dfeta(xsip,etap)
dfxsi = dNidxsi*fi(:)
dfeta = dNideta*fi(:)
```

- Exercise 1.3, Part 3:

Listing 7: code/Einheit1/EN/script_FEUE_5.m

```
%-----------------------------------------------------------------------
%  Script for the graphical illustration of the interpolation
%  functions dfdxsi and dfdeta with node values f1,f2,f3,f4
%
%  Copyright: IMWS
%  Version:   2 (EN)
%  Datum:     20151109
%-----------------------------------------------------------------------

clear
close all
clc

%% Enter node values
% f1.....value of f at node 1; f1 = f(xsi = +1,eta = +1)
% f2.....value of f at node 2; f2 = f(xsi = -1,eta = +1)
% f3.....value of f at node 3; f3 = f(xsi = -1,eta = -1)
% f4.....value of f at node 4; f4 = f(xsi = +1,eta = -1)
f1 = 5;
f2 = 1;
f3 = 3;
f4 = 0;

%% Computation
% Formation of node row vector fi
fi = [f1,f2,f3,f4];

% Enter value domains of xsi and eta as vectors
xsi = -1:0.1:1;
eta = -1:0.1:1;
lxsi = numel(xsi);
leta = numel(eta);

% Computation of function values within xsi = -1..+1,eta = -1..+1
% Initialize dfdxsi and dfdeta
dfdxsi = zeros(lxsi,leta);
dfdeta = zeros(lxsi,leta);
```

```
xsiMat = zeros(lxsi,leta);
etaMat = zeros(lxsi,leta);


for i = 1:lxsi
   for j = 1:leta
      % Evaluate derivations of the interpolation functions at
      % nodes xsi(i), eta(j)
      [dNidxsi,dNideta] = dshapefunc(xsi(i), eta(j));
      dfdxsi(i,j) = dNidxsi*fi(:);
      dfdeta(i,j) = dNideta*fi(:);

      % Save coordinates in additional matrix
      xsiMat(i,j) = xsi(i);
      etaMat(i,j) = eta(j);
   end
end

% Generate new figure
figure

% Graphical illustration of the interpolation function dfdxsi
surf(xsiMat,etaMat,dfdxsi)

% Generate new figure
figure

% Graphical illustration of the interpolation function dfdeta
surf(xsiMat,etaMat,dfdeta)
```

- Exercise 1.4, Part 1:

Listing 8: code/Einheit1/EN/jacobi.m

```
function[J] = jacobi(xi,yi,dNidxsi,dNideta)
%--------------------------------------------------------------------
%   Function for the computation of the Jacobi-Matrix (J-Matrix)
%   at coordinates (xsi,eta)
%
%   Input:  dNidxsi...dNi(xsi,eta)/dxsi =
%                     [dN1/dxsi,dN2/dxsi,dN3/dxsi,dN4/dxsi]
%           dNideta...dNi(xsi,eta)/deta =
%                     [dN1/deta,dN2/deta,dN3/deta,dN4/deta]
%           xi.......xi = [x1,x2,x3,x4]; x-coordinate of nodes
%           yi.......yi = [y1,y2,y3,y4]; y-coordinate of nodes
%
%   Output: J........J = [ dx/dxsi, dy/dxsi;
%                          dx/deta, dy/deta ]
%                     Jacobi-Matrix
%
%   Syntax:
%   J = jacobi(xi,yi,dNixsi,dNieta)
%
%   Copyright: IMWS
%   Version:   2 (EN)
%   Datum:     20151109
%--------------------------------------------------------------------

% Computation of derivations at coordinates (xsi,eta)
% dxdxsi = dx(xsi,eta)/dxsi
dxdxsi = dNidxsi*xi(:);

```

```
% dydxsi = dy(xsi,eta)/dxsi
dydxsi = dNidxsi*yi(:);

% dxdeta = dx(xsi,eta)/deta
dxdeta = dNideta*xi(:);

% dydeta = dy(xsi,eta)/deta
dydeta = dNideta*yi(:);

% Fit into J-Matrix
J = [ dxdxsi, dydxsi; ...
      dxdeta, dydeta];

end
```

- Exercise 1.4, Part 2:

Listing 9: code/Einheit1/EN/dxyshapefunc.m

```
function[dNidx,dNidy] = dxyshapefunc(J,dNidxsi,dNideta)
%----------------------------------------------------------------------
%  Function for the computation of values dNi/dx resp. dNi/dy
%  (i = 1,2,3,4) at coordinates (xsi,eta)
%
%  Input:  dNixsi...dNi(xsi,eta)/dxsi =
%                   [dN1/dxsi,dN2/dxsi,dN3/dxsi,dN4/dxsi]
%          dNieta...dNi(xsi,eta)/deta =
%                   [dN1/deta,dN2/deta,dN3/deta,dN4/deta]
%          J........J = [dx/dxsi,dy/dxsi;dx/deta,dy/deta]
%                   Jacobi-Matrix
%
%  Output: dNix.....dNi/dx = [dN1/dx,dN2/dx,dN3/dx,dN4/dx]
%          dNiy.....dNi/dy = [dN1/dy,dN2/dy,dN3/dy,dN4/dy]
%
%  Syntax:
%  [dNidx,dNidy] = dxyshapefunc(J,dNidxsi,dNideta)
%
%  Copyright: IMWS
%  Version:   2 (EN)
%  Datum:     20151109
%----------------------------------------------------------------------

% Solve linear equation system [dNdxsi; dNdeta] = J*[dNdxsi; dNdeta]
dN = J\[dNidxsi; dNideta];

% Split in dNdxsi and dNdeta
dNidx = dN(1,:);
dNidy = dN(2,:);
end
```

- Exercise 1.4, Part 3:

Listing 10: code/Einheit1/EN/script_FEUE_6.m

```
%----------------------------------------------------------------------
%  Script for the graphical illustration of the interpolation function
%  and their derivations dfdx and dfdy with node values f1,f2,f3,f4
%  and given coordinates x1,y1,x2,y2,x3,y3,x4,y4 within the x,y-
%  system
%
%  Copyright: IMWS
%  Version:   2 (EN)
%  Datum:     20151109
```

```
%-----------------------------------------------------------------------

clear
close all
clc

%% Enter node values
% f1.....value of f at node 1; f1 = f(xsi = +1,eta = +1)
% f2.....value of f at node 2; f2 = f(xsi = -1,eta = +1)
% f3.....value of f at node 3; f3 = f(xsi = -1,eta = -1)
% f4.....value of f at node 4; f4 = f(xsi = +1,eta = -1)
f1 = 0;
f2 = 1;
f3 = 2;
f4 = 1;

% node 1 (x1,y1) -> (xsi = +1,eta = +1)
% node 2 (x2,y2) -> (xsi = -1,eta = +1)
% node 3 (x3,y3) -> (xsi = -1,eta = -1)
% node 4 (x4,y4) -> (xsi = +1,eta = -1)
x1 = 3;
y1 = 1;

x2 = 1;
y2 = 4;

x3 = -1;
y3 = 4;

x4 = 1;
y4 = -3;

%% Computation
% Formation of a node row vector fi
fi = [f1,f2,f3,f4];
% Fit in node values
xi = [x1,x2,x3,x4];
yi = [y1,y2,y3,y4];

% Enter value domains of xsi and eta as vectors
xsi = -1:0.1:1;
eta = -1:0.1:1;
lxsi = numel(xsi);
leta = numel(eta);

% Computation of function values within xsi = -1..+1,eta = -1..+1
% Initialize...
f = zeros(lxsi,leta);
dfdx = zeros(lxsi,leta);
dfdy = zeros(lxsi,leta);
xMat = zeros(lxsi,leta);
yMat = zeros(lxsi,leta);

for i = 1:lxsi
   for j = 1:leta
      % Evaluate derivation of interpolation functions
      % nodes xsi(i), eta(j) with respective to xsi and eta
      [dNidxsi,dNideta] = dshapefunc(xsi(i), eta(j));

      % Evaluate derivation of interpolation functions
```

```
        % nodes xsi(i), eta(j) with respective to x and y
        J = jacobi(xi,yi,dNidxsi,dNideta);
        [dNdx,dNdy] = dxyshapefunc(J,dNidxsi,dNideta);
        % Evaluate derivations of function f
        dfdx(i,j) = dNdx*fi(:);
        dfdy(i,j) = dNdy*fi(:);

        Ni = shapefunc(xsi(i),eta(j));
        f(i,j) = Ni*fi(:);

        % Save coordinates in additional matrices xsiMat and
        % etaMat
        xMat(i,j) = Ni*xi(:);
        yMat(i,j) = Ni*yi(:);
    end
end

% Generate new figure
figure

% graphical illustration of interpolation function f
surf(xMat,yMat,f)

% Generate new figure
figure

% graphical illustration of interpolation function dfxsi
surf(xMat,yMat,dfdx)

% Generate new figure
figure

% graphical illustration of interpolation function dfeta
surf(xMat,yMat,dfdy)
```