Übung 1: Tensorrechnung und Kontinuumsmechanik

Aufgabe 1.1: Tensormultiplikation

Schreiben Sie die folgenden Ausdrücke in Indexnotation und vereinfachen Sie soweit es geht.

a) $\boldsymbol{I} : (\boldsymbol{A} \otimes \boldsymbol{b})$

b) $(\boldsymbol{I} \otimes \boldsymbol{I})^{T23} : \boldsymbol{B}$

c) $\boldsymbol{B} : (\boldsymbol{A} \otimes \boldsymbol{A}^T) : \mathbb{L} : (\boldsymbol{B} \otimes \boldsymbol{B}^T) : \boldsymbol{A}$

d) $\boldsymbol{I}\,(\boldsymbol{A}\,\boldsymbol{B})$

e) $(\boldsymbol{I} \otimes \boldsymbol{I}) : \boldsymbol{B}$

f) $\boldsymbol{A}^T : \mathbb{A} : \boldsymbol{B}$

Aufgabe 1.2: Differentialoperationen

Schreiben Sie die folgenden Ausdrücke in Indexnotation und vereinfachen Sie soweit es geht.

a) $\frac{1}{2}(\operatorname{grad} \boldsymbol{v} + \operatorname{grad}^T \boldsymbol{v}) : \boldsymbol{A}$ (Ann.: $\boldsymbol{A} = \boldsymbol{A}^T$)

b) $\operatorname{div}(\boldsymbol{v}\,\boldsymbol{T})$

c) $\dfrac{\partial(\operatorname{tr} \boldsymbol{A})}{\partial \boldsymbol{A}}$

d) $\dfrac{\partial \boldsymbol{A}}{\partial \boldsymbol{A}} : \boldsymbol{B}$

e) $\operatorname{tr}(\operatorname{grad}(\boldsymbol{v}\,\boldsymbol{T}))$

f) $\dfrac{\partial(\boldsymbol{A}\,\boldsymbol{B})}{\partial \boldsymbol{A}}$

Aufgabe 1.3: Elastizitätstensor und Voigt-Notation

Für linear elastisches isotropes Material lässt sich der Elastizitätstensor $\mathbb{C}$ wie folgt darstellen:

$$\mathbb{C} = \lambda\,\boldsymbol{I} \otimes \boldsymbol{I} + 2\mu\,\mathbb{I} \tag{1.1}$$

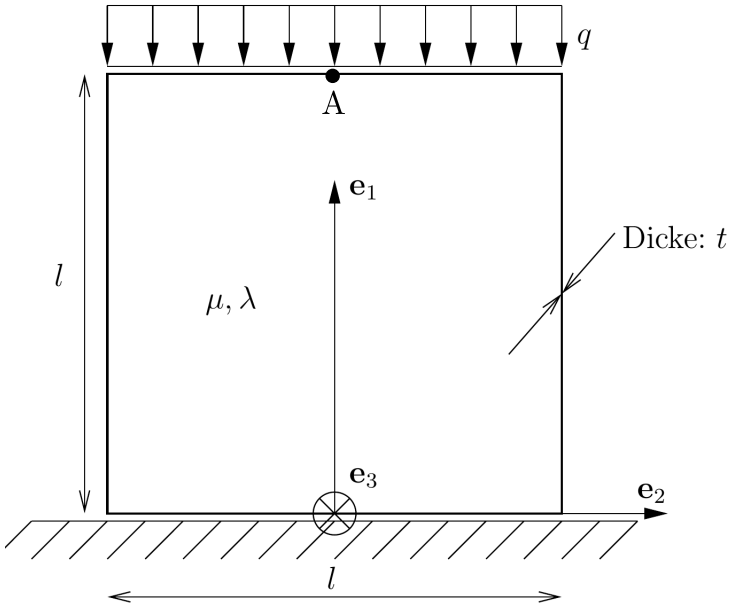
a) Zeigen Sie, dass die Ausdrucksweisen $\boldsymbol{\sigma} = \lambda\,(\operatorname{tr} \boldsymbol{\varepsilon})\,\boldsymbol{I} + 2\mu\,\boldsymbol{\varepsilon}$ und $\boldsymbol{\sigma} = \mathbb{C} : \boldsymbol{\varepsilon}$ äquivalent sind. Setzen Sie dazu (1.1) ein, schreiben die beiden Ausdrücke in Indexschreibweise und vereinfachen Sie soweit wie möglich.

b) Schreiben Sie das Materialgesetz $\boldsymbol{\sigma} = \mathbb{C} : \boldsymbol{\varepsilon}$ zunächst in genereller Matrixnotation und dann in Voigt-Notation. Unter welchen Annahmen ist dies möglich?

Hinweis: $\boldsymbol{\varepsilon}$ lautet in Voigt-Notation $\underline{\boldsymbol{\varepsilon}}^V = [\varepsilon_{11},\, \varepsilon_{22},\, \varepsilon_{33},\, 2\varepsilon_{23},\, 2\varepsilon_{13},\, 2\varepsilon_{12}]^T$

Aufgabe 1.4: Analytische Lösung eines Randwertproblems

Der dargestellte Körper $\mathcal{B} = l \times l \times t$ liegt reibungsfrei auf einer Ebene und ist im Koordinatenursprung (mittig in der l-t-Ebene) zusätzlich in Horizontalrichtung fixiert. Auf ihn wirkt die Flächenlast q. Volumenkräfte durch die Erdbeschleunigung sind zu vernachlässigen.



a) Erfüllt der Verschiebungszustand $\boldsymbol{u} = \alpha x_1 \boldsymbol{e}_1 + \beta x_2 \boldsymbol{e}_2 + \gamma x_3 \boldsymbol{e}_3$ die Impulsbilanz? Bestimmen Sie die Konstanten α, β, γ aus den Randbedingungen. Nutzen sie dazu das Cauchy-Theorem $\boldsymbol{\sigma}\,\boldsymbol{n} = \boldsymbol{t}$

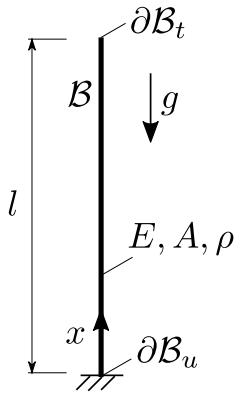
b) Berechnen Sie die Verschiebung im Punkt A für den Parametersatz $\{E, \nu, q, t, l\} = \{210\text{GPa}, 0.3, 500\text{MPa}, 100\text{mm}, 100\text{mm}\}$

Hinweis: Die Beziehungen zwischen den Elastizitätsparametern lauten:

$$\lambda = \frac{E\nu}{(1+\nu)(1-2\nu)} \quad , \quad \mu = \frac{E}{2(1+\nu)}$$

Übung 2: Variationsrechnung

Aufgabe 2.1: Stab unter Eigengewicht



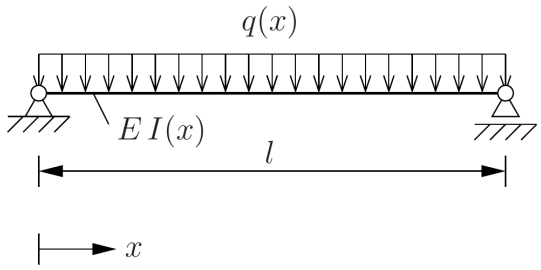
Ein Stab ($EA = const.$) der Dichte ρ und der Länge l ist eingespannt an der Position $x = 0$ und erfährt eine Deformation aufgrund seiner Masse im Erdschwerefeld g.

a) Leiten Sie mithilfe des Prinzips des Minimums des elastischen Gesamtpotentials die Euler-Lagrange Differentialgleichung für die Verschiebungsfunktion $u(x)$ her.

b) Ermitteln sie die Lösungsfunktion $u(x)$ der Differentialgleichung unter Berücksichtigung der Randbedingungen.

Aufgabe 2.2: Balken unter Streckenlast

Ein Balken ($EI = const.$) der Länge l erfährt eine konstante Streckenlast $q(x) = q_0$.



Vorgegeben sei das folgende elastische Gesamtpotential:

$$\Pi = \int_0^l \left[\frac{1}{2} EI(w(x)'')^2 - w(x)q_0\right] \mathrm{d}x$$

a) Leiten Sie mithilfe der Variationsmethode die Euler-Lagrange Gleichung für die Durchbiegung $w(x)$ sowie die natürlichen Randbedingungen her.

b) Bestimmen sie die Biegelinie $w(x)$ unter Auswertung der Randbedingungen

c) Wie lauten die natürlichen Randbedingungen unter der Annahme, dass der Balken bei $x = 0$ fest eingespannt ist und bei $x = l$ frei?

Hinweis: Die essentiellen RB lauten in diesem Fall $w(0) = 0$ und $w'(0) = 0$.

Übung 3: Einführung Matlab

Einführung

Was ist MATLAB **?**

- MATLAB (abkürz. für MATrix LABoratory) ist ein Software-Paket für numerische Berechnung und Visualisierung. Es besteht aus einer großen Anzahl an vorgefertigten Funktionen (z.B. lineare Algebra, Lösen von Differentialgleichungen, ...)
- Bei MATLAB steht der Fokus mehr auf numerischer Berechnung, während die Funktionalität von MAPLE und MATHEMATICA beispielsweise mehr auf symbolische Algebera ausgerichtet ist.
- Bekannte MATLAB-ähnliche Open-Source Programme: SCILAB, OCTAVE

Grund-Features:

- Automatische Dimensionierung (d.h., keine Dimensionsangaben bei Deklaration von Vektoren und Arrays erforderlich)
- Case-Sensitive (d.h., `a` und `A` sind unterschiedliche Variablen)
- Eingebaute Funktionen basieren auf Vektor- und Matrixoperationen und sind dementsprechend auch für diese optimiert.
- Datei-Typen:
  - M-Dateien: Standard ASCII Textdateien mit `.m` -Dateiendung
  - Mat-Dateien: Binärdateien, mit `.mat` -Dateiendung
  - Mex-Dateien ("MATLAB-Executable"): C, C++ oder Fortran Programme, welche von MATLAB aufgerufen werden können
- Grundlegendes Daten-Objekt ist das Array, welches aus Unterelementen verschiedenster Typen (Integern, Doubles, Matrizen, Characters, Strings, Strukturen und Zellen) bestehen kann.
- Der Typ einzelner Daten-Objekte muss nicht explizit deklariert werden. (z.B. besteht auch keine Notwendigkeit Variablen als Reel oder Komplex zu deklarieren)

Grundbefehle

Hilfe

`help <command>`	Hilfe zu `<command>`
`lookfor <string>`	listet Hilfe zu `<string>` auf

Verzeichnisbefehle

`pwd`	aktuelles Verzeichnis anzeigen
`cd`	Verzeichnis ändern
`dir` / `ls`	Inhalt des aktuellen Verzeichnisses anzeigen

Beispiel:

```
>> help cos
 COS    Cosine.
    COS(X) is the cosine of the elements of X.
 Overloaded methods
    help sym/cos.m
```

Variablen

Operatoren

`=`	einen Wert zuweisen
`+ - * / ^`	Rechenoperatoren
`,` (am Ende der Zeile)	Befehl mit Output
`;` (am Ende der Zeile)	kein Output
`...`	Zeilenumbruch innerhalb eines Befehls

Konstanten

`i,j`	Imaginärwert $\sqrt{-1}$
`pi`	$\pi = 3.14\ldots$
`inf`	Unendlich inf
`NaN`	not a Number, z. B. $\frac{0}{0}$
`eps`	kleinste positive Zahl, welche sich ausgeben lässt

Variablenmanagement

`who`	listet die Namen aller im Workspace befindlichen Variablen auf
`whos`	listet Namen und Größe der Variablen
`clear`	clear Workspace
`clear <var>`	clear Variable `<var>`
`clc` / `clf` / `cla`	clear command / figure / axes

Für mehr Informationen: Eingabe von `help +` in MATLAB-Kommandozeile.

Beispiele:

```
>> ( 47 + 1e+02 * 1.5 + 4^2 ) / 4
ans =
   53.2500
```

Die Variable `ans` ist das Ergebnis der letzten Rechenoperation.

```
>> a = ( 47 + 1e+02 * 1.5 + 4^2 ) / 4
a =
   53.2500
```

Das Ergebnis wird in Variable `a` gespeichert.

Mathematische Funktionen

`sqrt(x)`	Quadratwurzel
`xp(x)`	Exponentialfunktion
`log(x)` / `log10`	Logarithmusfunktionen
`sin(x)`	Sinus
`cos(x)`	Cosinus
`tan(x)`	Tangens
`atan(x)`	Arkustangens (Winkel $-90° \ldots +90°$)
`atan2(y,x)`	Arkustangens (Winkel $-180° \ldots +180°$)
`abs(x)`	Betrag von `x`
`sign(x)`	Signum (Vorzeichen von `x`)

Mehr Infos: Eingabe von `help elfun` oder `help datafun` in MATLAB-Kommandozeile.

Beispiel:

```
>> y1 = 2^2+log(pi)*sin(0.75*pi/2)+sqrt(exp(2*pi/3))

y1 =
    7.9072
```

$$y_1 = 2^2 + \ln \pi \sin(0.75\pi/2) + \sqrt{e^{2/3\pi}}$$

Vektoren und Matrizen

Vektor- und Matrixbefehle

`[ x1 x2 ...; y1,y2,...]`	Vektor oder Matrix (',' oder Leerzeichen zwischen Spalten, ';' oder Zeilenumbruch zwischen Reihen)
`start: <stepsize:> end`	Spaltenoperator (`stepsize` opt., sonst = 1)
`linspace(start,end,num_steps)`	linearer Zeilenvektor
`logspace(start,end,num_steps)`	logarithmischer Zeilenvektor
`eye(rows,columns)`	Einheitsmatrix [Zeilen x Spalten]
`ones(rows,columns)`	Matrix, bei der alle Einträge=1 sind [Zeilen x Spalten]
`zeros(rows,columns)`	Nullmatrix [Zeilen x Spalten]
`rand(rows,columns)`	Matrix mit Zufallswerten [Zeilen x Spalten]
`a(index)`	Element des Vektors `a` an Position `index`
`A(r,c)`	Element der Matrix `A` an Position `r` x `c`

Beispiele:

```
>> x=[1 2 3]
x =
     1     2     3
>> y=[4;5;6]
y =
     4
     5
     6
>> A=[1 2 3;4,5,6;7 8,9]
A =
     1     2     3
     4     5     6
     7     8     9
>> A(2,3)
ans =
     6
```

```
>> A(1,2)=8
A =
     1     8     3
     4     5     6
     7     8     9
>> B=A(2:3,1:3)
B =
     4     5     6
     7     8     9
>> v=0:2:8
v =
     0     2     4     6     8
>> v=[8:-2:0]
v =
     8     6     4     2     0
```

Operationen

`.* .\ ./ .^`	elementweise Berechnung
`\ /`	linke und rechte Division
`transpose(A)` oder `A.'`	Transponierte von `A`
`ctranspose(A)` or `A'`	Transponierte von `A` (komplex Konjugiert)
`inv(A)`	Inverse von `A`
`det(A)`	Determinante von `A`

Dimensionen

`[M,N] = SIZE(A)`	Dimension von Matrix und Vektor
`M = SIZE(A,DIM)`	Länge der Komponente `DIM` der Matrix `A`

Mathematische Funktionen von Vektoren und Matrizen

`sum(a)`	Summe der Vektorelemente
`prod(a)`	Produkt der Vektorelemente
`min(a)`	kleinstes Vektorelement
`max(a)`	größtes Vektorelement
`sort(a)`	Elemente in aufsteigender Reihenfolge
`find(a)`	nicht-Null Elemente

Beispiele:

```
>> A=[1 2 3; 4 5 6; 7 8 9];
>> B=[1 2 3; 2 4 5; 3 7 8];
>> b=[2 4 6 8 10]';
>> v=0:2:8;

>> v*b
ans =
   160
>> v'*b'
ans =
     0     0     0     0     0
     4     8    12    16    20
     8    16    24    32    40
    12    24    36    48    60
    16    32    48    64    80
```

```
>> c=v+b';
>> c=v-b';
>> C=A+B;
>> C=A-B;
>> C=A*B;
>> c=A*v(1:3)'
c =
    16
    34
    52
>> c=v.*b'
c =
     0     8    24    48    80
```

Beispiel:

```
>> A=[5 -3 2; -3 8 4; 2 4 -9];
>> b=[10;20;9];
>> x=A\b
x =
    3.4442
    3.1982
    1.1868
```

Lösen des linearen Gleichungssystems: $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$

$$\begin{aligned} 5x &= 3y - 2z + 10 \\ 8y + 4z &= 3x + 20 \\ 2x + 4y - 9z &= 9 \end{aligned}$$

Skripte und Funktionen

- Skript:
  Wenn ein Skript aufgerufen wird, führt MATLAB schlicht die in der Datei befindlichen Befehle aus. Skripte können im Workspace bereits vorhandene Daten bearbeiten oder auch neue Daten erzeugen. Auch wenn Skriptdateien keine Outputargumente aufweisen, werden alle erzeugten Variablen im Workspace gespeichert und können in darauffolgenden Rechenoperationen verwendet werden.

- Funktion:
  Funktionen sind `.m`-Dateien, welche Inputargumente aufnehmen und Outputargumente wiedergeben. Der Name der `.m`-Datei und der Funktion sollten übereinstimmen. Funktionen arbeiten mit lokalen Variablen in einem funktioneigenen Worskpace unabhängig des von der MATLAB-Kommandozeile bearbeiteten Workspaces.

Sowohl Skripte als auch Funktionen werden als .m-Datei gespeichert.

Struktur von Funktionen:

```
function [a_out,b_out] = name_of_function(a_in,b_in)
%         output list                     input list
%
% Description of function can be placed here
% by using the comment-operator '%'. This
% informations can be displyed by typing
% 'help name_of_function' at MATLAB-command-line.
.
.
a_out = a_in - a_out;
b_out = a_in + a_out;
.
.
```

Beispiel: Länge eines Vektors berechnen

```
function vlength = fvectorlength(vector)
%
% Computation of the length 'vlength' of
% a column vector 'vector'

vlength = sqrt(vector'*vector);
```

Aufrufen der Funktion:

```
>> fvectorlength([-1;3;5])
ans =
    5.9161
```

Logikoperatoren

`== , ~=`	gleich, nicht gleich		
`< , <=`	kleiner als, kleiner gleich		
`> , >=`	größer als, größer gleich		
`~`	logisches NOT	`&`	elementweise logiesches AND
`|`	elementweise logisches OR	`xor`	logisches EXCLUSIVE-OR

Mehr Infos: Eingabe von `help +` in MATLAB-Kommandozeile.

IF-/Fallunterscheidungen und Schleifen

`IF` expression (z.B. `a==1`)
  statements
`ELSEIF` expression
  statements
`ELSE`
  statements
`END`
	IF-Statement Bedingung
	
	ELSE und ELSEIF sind optional

`SWITCH` switch-expr (z.B. `id`)
  `CASE` case-expr, (z.B. `1`)
  statement, ..., statement
  `CASE {`case-expr1, case-expr2, case-expr3,...`}`
  statement, ..., statement
  `OTHERWISE,`
  statement, ..., statement
`END`
	SWITCH-Statement Fall

`FOR` variable = expr (z.B. `ii=1:10`) statement, ..., statement `END`	Wiederholung der Statements in einer spezifischen Anzahl.
`WHILE` expression (z.B. `ii<imax`) statements `END`	Wiederholung der Statements eine unbestimmte Anzahl bis Bedingnung erfüllt.

Plotten von Schaubildern

Bedienen des 'figure'-Fensters

`figure , figure(no)`	erstellen, aktivieren eines Schaubilds mit Nummer `no`
`subplot(num_rows,num_cols,index)`	erstellen eines Subplots z.B., `subplot(2,3,2)`: Subplot mit 2 Zeilen und 3 Spalten, aktivieren eines 2. Subplots
`gcf`	'get current figure'
`clf`	'clear active figure'
`delete(id)`	Objekte mit `id` löschen
`close(index)`	'figure'-Fenster Nr. `index` löschen
`close all`	alle 'figure'-Fenster löschen
`hold <on | off>`	aktuelles 'figure'-Fenster geöffnet lassen bei Generierung neuer Plots an/aus

2-D Plot Befehle

`plot(<x,>y<,plotstyle>,...)`	Plot, Linear (`<>` = opt.)
`fplot(func,range)`	Plot einer expliziten Funktion
`line(x,y)`	erstellen der Linie durch Koordinatenvektoren `x,y`
`text(x,y,string)`	platzieren von `string` an Position `x,y`
Für mehr Infos über 2-D Plots: aufrufen von `help plot` in Kommandozeile.	

3-D Plot Befehle

`plot3(x,y,z <,plotstyle> ,...)`	dreidimensionaler Plot
`sphere`	plotten einer Kugel
`[x,y,z] = sphere`	Koordinaten einer Einheitskugel unter `x,y,z` speichern
`line(x,y,z)`	erstellen einer Linie durch Koordinatenvektoren `x,y,z`
`text(x,y,z,string)`	platzieren von `string` an Position `x,y,z`

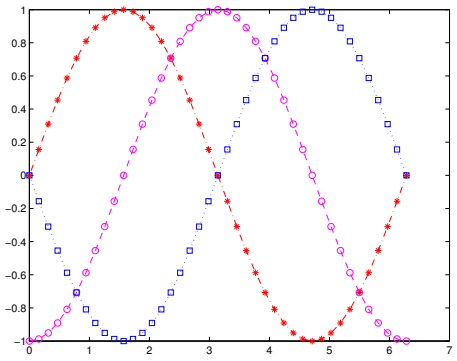
Plotstyles

Farben:			Linienstyles:		
`k` schwarz	`r` rot	`g` grün	`-` durchgängig	`o` Kreis	`.` Punkte
`b` blau	`m` magenta	`w` weiß	`--` gestrichelt	`*` sterne	`x` x
`c` cyan	`y` gelb		`:` gepunktet	`+` plut	`-.` Punkt-Strichlinie

Mehr Infos: `help plot`

Beispiele:

```
>> figure(1)
>> clf
>> t=0:pi/20:2*pi;
>> plot(t,sin(t),'-.r*')
>> hold on
>> plot(t,(sin(t-pi/2)),'linestyle','--',...
   'marker','o','color','m')
>> plot(t,sin(t-pi),':bs')
>> hold off
```

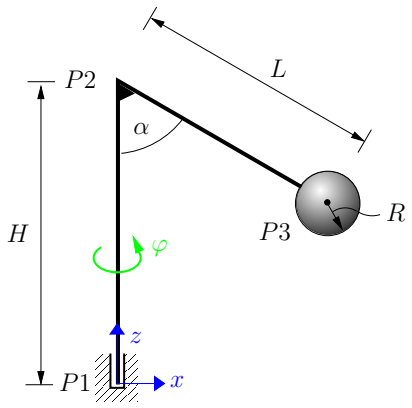


Labeln von Achsen und Figures

`axis([xmin,xmax,ymin,ymax<,zmin,zmax>)`	Skalierung (min=`-inf` oder max=`inf` → Auto Skalierung)
`axis <on | off | auto | equal | square>`	verschiedene Achsen-Befehle Mehr: `help axis`
`grid <on | off>`	Gitter an/aus
`gca`	Wiedergabe der aktuellen Achse
`cla`	löschen der aktuellen Achse
`xlabel(string)`	Label der x-Achse
`ylabel(string)`	Label der y-Achse
`zlabel(string)`	Label der z-Achse
`title(string)`	Titel der aktuellen Achse
`legend(string_1,string_2,...<,pos>)`	Legende platzieren `pos = 0,1,2,3,4,-1`

Aufgabe 3.1: Animation eines rotierenden Kragarms

Ein Kragarm der Länge L ist am oberen Ende (Punkt P2) an einer vertikalen Stange (höhe H) befestigt. Der Kragarm ist im Winkel α zur Stange geneigt. Die Stange ist am unteren Ende (Punkt P1) drehbar gelagert mit der z-Achse als Rotationsachse. Am Freien Ende des Kragarms (Punkt P3) ist eine Kugel mit dem Radius R befestigt.



Programmieren Sie ein MATLAB-Skript (`.m`-file), in welchem die Rotation φ um die z-Achse des dargestellten Systems in animierter Form dargestellt wird.

Hinweis: Nutzen sie die Rotationsmatrix

$$\boldsymbol{R} = \begin{bmatrix} \cos\varphi & -\sin\varphi & 0 \\ \sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

welche mit $\bar{\boldsymbol{v}} = \boldsymbol{R}\boldsymbol{v}$ die Rotation eines Vektors $\boldsymbol{v}$ um die z-Achse beschreibt.

Aufgabe 3.2: Animation mithilfe einer Subroutine

Das MATLAB-Skript aus der vorherigen Aufgabe soll nun modifiziert werden. Eine selbst programmierte Funktion soll die rotierten Koordinaten für die Animation berechnen. Input parameter sollen die die Ursprungskoordinaten und der Rotationswinkel φ sein. Der Output der Funktion sollten die Koordinaten des rotierten Systems sein.

Übung 4: Assemblierung

Aufgabe 4.1: Ebenes Fachwerk

Gegeben sei das folgende Fachwerk:

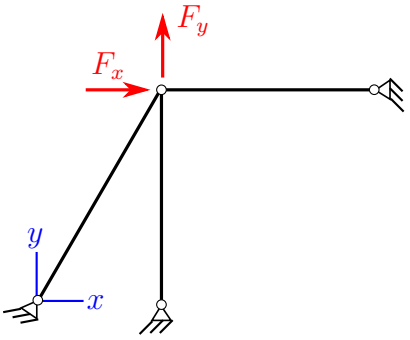


Abbildung 4.1: Randwertproblem

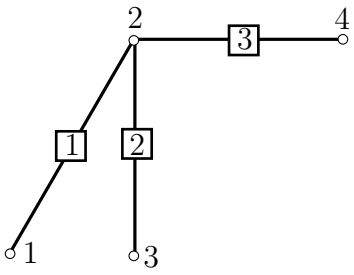


Abbildung 4.2: Element- und Knotennummerierung

a) Welche grundlegenden Schritte sind erforderlich um das Problem mithilfe der Finite-Elemente Methode zu lösen?

Im Folgenden sei angenommen, dass die jeweiligen Elementsteifigkeitsmatrizen $\underline{\boldsymbol{k}}^e$ bekannt sind.

b) Assemblieren Sie aus den Elementsteifigkeitsmatrizen die globale Steifigkeitsmatrix $\underline{\boldsymbol{K}}$ unter Berücksichtigung der Dirichlet-Randbedingungen. Stellen Sie den globalen Lastvektor $\underline{\boldsymbol{P}}$ und das globale Gleichungssystem auf.

Nutzen Sie die in Abbildung 4.2 dargestellte Element- und Knotennummerierung.

Aufgabe 4.2: Ebenes Fachwerk

Gegeben sei das dargestellte ebene Fachwerk, für das die jeweiligen Elementsteifigkeitsmatrizen $\underline{\boldsymbol{k}}^e$ als bekannt angenommen werden.

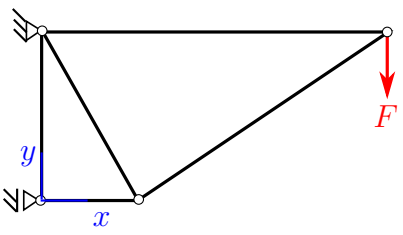


Abbildung 4.3: Randwertproblem

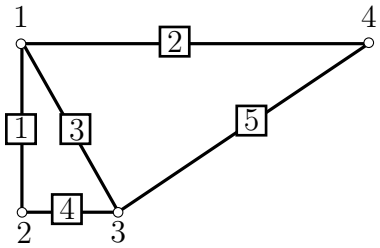


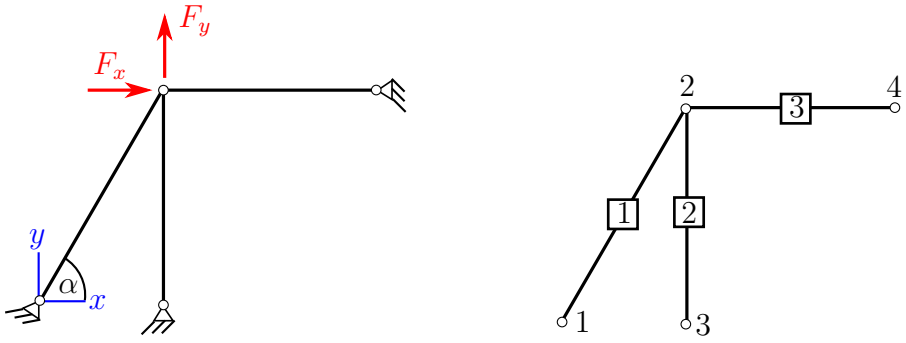
Abbildung 4.4: Diskretisierung

Stellen Sie den globalen Lastvektor und die globale Steifigkeitsmatrix unter Berücksichtigung der Dirichlet-Randbedingungen auf. Nutzen Sie die vorgegebene Diskretisierung.

Übung 5: FEM mit Stabelementen

Aufgabe 5.1: Elementsteifigkeitsmatrix

Betrachtet wird das Fachwerk aus Aufgabe 4.1.



a) Wie lassen sich die Einträge der Elementsteifigkeitsmatrizen $\underline{\boldsymbol{k}}^e$ berechnen? Werten Sie auch die Spezialfälle $\alpha \in \{\pi/2, 3\pi/2\}$ und $\alpha \in \{0, \pi\}$ aus.

b) Schreiben Sie die MATLAB-Funktion `PlaneTrussElementStiffness.m` zur Berechnung der Elementsteifigkeitsmatrizen. Verwenden sie die Parameter aus Tabelle 5.1

Hinweis: `PlaneTrussElementStiffness.m` ist eine Subroutine zur Berechnung der Elementsteifigkeitsmatrix im globalen Koordinatensystem. Die Inputwerte sind E-Modul `E`, Stabquerschnitt `A` und die Knotenkoordinaten `i` und `j` (vgl. Abbildung 5.1).

```
node_i(1) = x(1)
node_i(2) = y(1)
node_j(1) = x(2)
node_j(2) = y(2)
```

Outputgröße ist die Elementsteifigkeitsmatrix im globalen x,y-Koordinatensystem:

$$\underline{\boldsymbol{k}}^{\mathrm{e}} = \underline{\boldsymbol{T}}^T \, \underline{\tilde{\boldsymbol{k}}}^{\mathrm{e}} \, \underline{\boldsymbol{T}} \quad \text{mit} \quad \underline{\tilde{\boldsymbol{k}}}^{\mathrm{e}} = \int_{\mathcal{B}^{\mathrm{e}}} (\underline{\boldsymbol{B}}^{\mathrm{e}})^T EA \, \underline{\boldsymbol{B}}^{\mathrm{e}} \, \mathrm{d}\tilde{x}$$

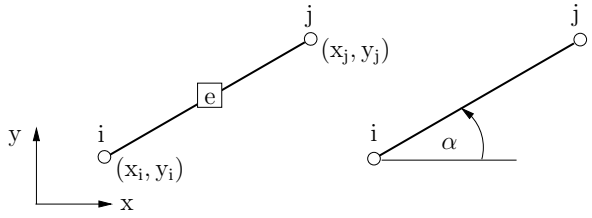


Abbildung 5.1

Inputparameter

`E`	`10000` kN/m^2
`A`	`0.02` m^2
`node no.1: [ x y ]`	`[ 0 0 ]` m
`node no.2: [ x y ]`	`[ 1 2 ]` m
`node no.4: [ x y ]`	`[ 3 2 ]` m

Tabelle 5.1

c) Bestimmen Sie den Lösungsvektor $\underline{\boldsymbol{D}}$ mithilfe des in der vorherigen Übung aufgestellten globalen Gleichungssystems. Die Kräfte betragen $F_x = 0.01\,\mathrm{kN}$ und $F_y = -0.05\,\mathrm{kN}$.

Aufgabe 5.2: Schreiben eines FEM-Programms

Im Folgenden soll ein eigener FE-Code in MATLAB erzeugt werden, mithilfe dessen die Knotenverschiebungen von ebenen Fachwerken berechnet werden können. Der Workflow des Programms kann Abbildung 5.2 entnommen werden. Dabei stellt die Datei `PlaneTruss.m` die Hauptdatei dar. Mit ihr werden zunächst Element- und Knotendaten des Fachwerkes eingelesen. Daraufhin werden mithilfe der Subroutinen `PlaneTrussElementStiffness.m` und `PlaneTrussAssemble.m` die Elementsteifigkeitsmatrizen aufgestellt und im globalen Gleichungssystem assembliert. Schließlich wird der globale Lastvektor aufgestellt und das globale Gleichungssystem gelöst.

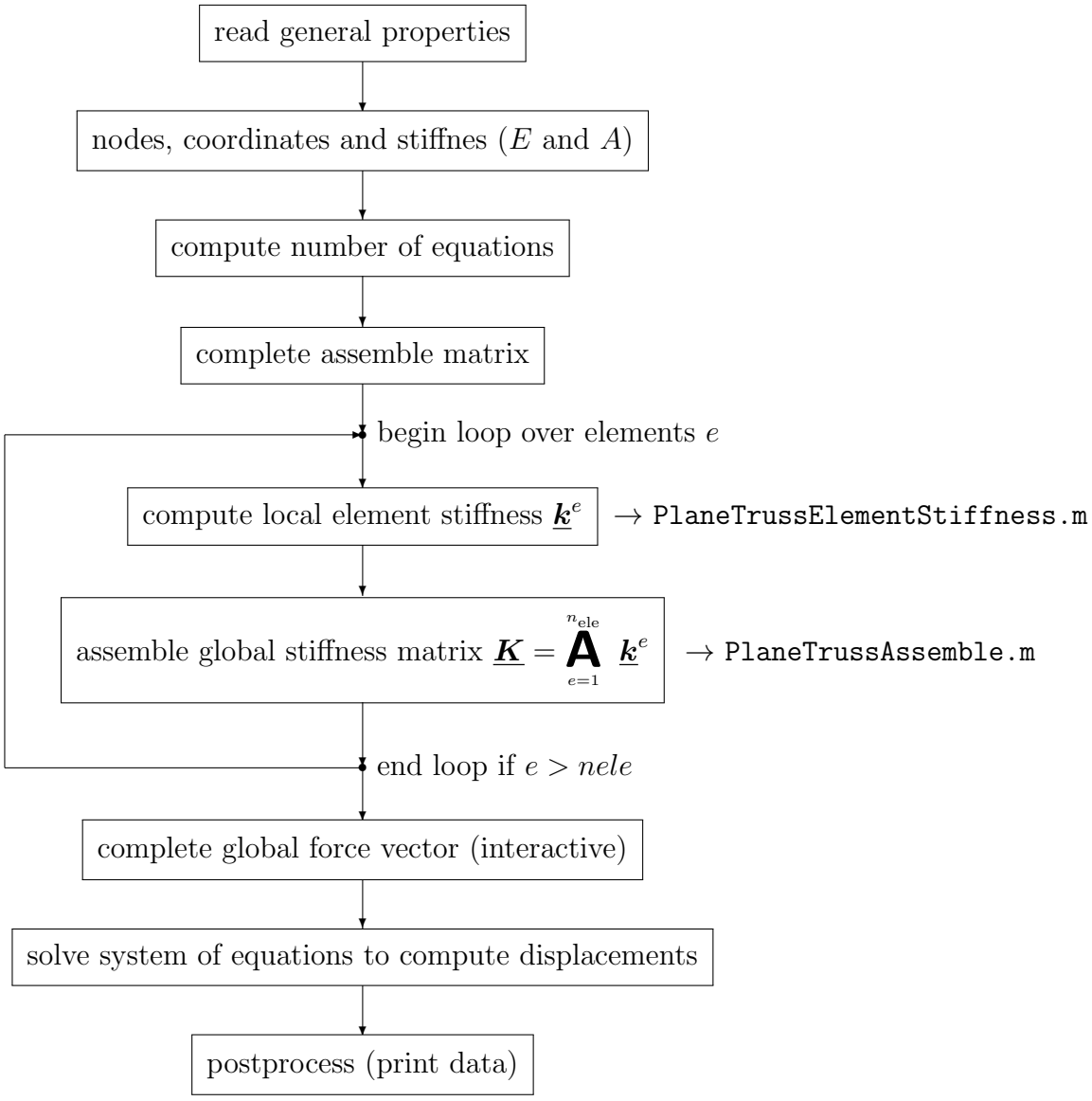


Abbildung 5.2: Workflow von `PlaneTruss.m`

a) Erzeugen sie zunächst in der Hauptdatei `PlaneTruss.m` ein Interface, mit welchem die Element- und Knotendaten sowie die Dirichlet-Randbedingungen eingelesen werden.

Hinweis: Anhand des einfachen Fachwerks in Abbildung 5.3 soll veranschaulicht werden wie die entsprechenden Daten eingelesen und abgespeichert werden.

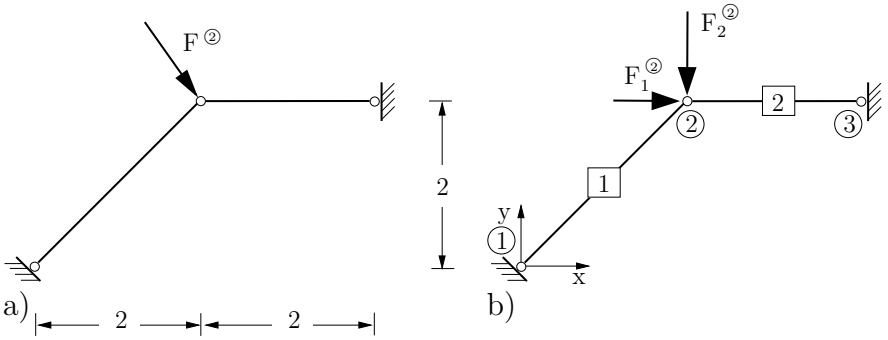


Abbildung 5.3: Fachwerk: a) Dimension (in Metern) und Randbedingungen, b) Diskretisierung.

Nach dem Programmstart soll zunächst folgender Input erfolgen:

```
>> PlaneTruss
give the total number of elements. num_ele = 2
give the total number of nodes.  num_nodes = 3
```

Im nächsten Schritt wird die Matrix `connectivity` erstellt. In ihr wird abgespeichert, welche globale Knotennummer zu welchem element gehört. Sie hat die Dimension [`nodes_ele` × `num_ele`], wobei `nodes_ele` = 2 die Anzahl der Knoten je Element darstellt. In diesem Beispiel besitzt Element 1 die globalen Knoten 1 und 2; Element 2 hat die Knoten 2 und 3. Die `connectivity`-Matrix sieht wie folgt aus:

$$\texttt{connectivity} = \begin{bmatrix} 1 & 2 \\ 2 & 3 \end{bmatrix}$$

Der E-Modul sei mit $E = 10000\,\mathrm{kN/m^2}$ festgelegt und die Querschnittsfläche der Elemente 1 und 2 sei

$$0.02\,\mathrm{m^2} \quad \text{und} \quad 0.015\,\mathrm{m^2}.$$

Die Elementdaten werden nun wie folgt eingegeben

```
element no.1
  global node for 1st local node = 1
  global node for 2nd local node = 2
  element data           [ E A ] = [10000 0.02]

element no.2
  global node for 1st local node = 2
  global node for 2nd local node = 3
  element data           [ E A ] = [10000 0.015],
```

wobei die eingegebenen Knotennummern in der `connectivity`-Matrix gespeichert werden.

Nun werden für alle Knoten die (x, y)-Koordinaten eingegeben Die entsprechende `X`-Matrix der Knotenkoordinaten der Dimension [`dof_nodes` × `num_nodes`] lautet:

$$\mathtt{X} = \begin{bmatrix} 0 & 2 & 4 \\ 0 & 2 & 2 \end{bmatrix}$$

Der Input sieht wie folgt aus:

```
node no.1
  global node coordinates [ x y ] = [0 0]

node no.2
  global node coordinates [ x y ] = [2 2]

node no.3
  global node coordinates [ x y ] = [4 2]
```

Die Anzahl an Dirichlet-Randbedingungen `num_bc` beträgt 4. Nach der Eingabe

```
 give the total number of displacement boundary conditions
                                          num_bc = 4
```

kann nun die Anzahl an verbleibenden Freiheitsgraden mit

$$\mathtt{num\_eq} = \mathtt{num\_nodes} * \mathtt{dof\_nodes} - \mathtt{num\_bc} \tag{5.1}$$

bestimmt werden. Die Horizontal- und Vertikalverschiebungen der Knoten 1 und 3 werden auf Null gesetzt:

$$\mathrm{d}_1^{①} = \mathrm{d}_2^{①} = \mathrm{d}_1^{③} = \mathrm{d}_2^{③} = 0,$$

bzw.

$$\mathrm{D}_1 = \mathrm{D}_2 = \mathrm{D}_5 = \mathrm{D}_6 = 0.$$

Im Zusammenhang mit diesem Schritt wird die Matrix `assembleid` erstellt. Sie hat die Dimension [`dof_nodes` × `num_nodes`] und beinhaltet alle verbleibenden Freiheitsgrade. An den Positionen, an denen Dirichlet-Randbedingungen vorliegen, befinden sich Nulleinträge. Für das Beispiel lautet sie also:

$$\mathtt{assembleid} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 2 & 0 \end{bmatrix}$$

Die Inputsequenz dazu lautet:

```
    input of global dofs with zero boundary displacements
                                  for i from 1 to num_bc
        global node of constrained degree of freedom = 1
          degree of freedom to constrained (1=x,2=y) = 1
```

```
global node of constrained degree of freedom = 1
  degree of freedom to constrained (1=x,2=y) = 2
global node of constrained degree of freedom = 3
  degree of freedom to constrained (1=x,2=y) = 1
global node of constrained degree of freedom = 3
  degree of freedom to constrained (1=x,2=y) = 2
```

b) Setzen Sie die Assemblierung der Elementsteifigkeitsmatrizen zur globalen Steifigkeitsmatrix algorithmisch in Form von MATLAB Code in der Hauptdatei `PlaneTruss.m` und mithilfe der Subroutinen `PlaneTrussElementStiffness.m` und `PlaneTrussAssemble.m` um.

Hinweis: Die Assemblierung findet in einer Schleife über die Elemente statt, in der für jedes Element zunächst die lokale Steifigkeitsmatrix $\underline{\boldsymbol{k}}^e$ (mithilfe von `PlaneTrussElementStiffness.m`) erzeugt wird. Mithilfe der Subroutine `PlaneTrussAssemble.m` soll die globale Steifigkeitsmatrix nun an den passenden Einträgen mit den jeweiligen Einträgen der lokalen Steifigkeitsmatrix besetzt werden. Um diesen Prozess zu veranschaulichen wird das Fachwerk in Abbildung 5.4 betrachtet

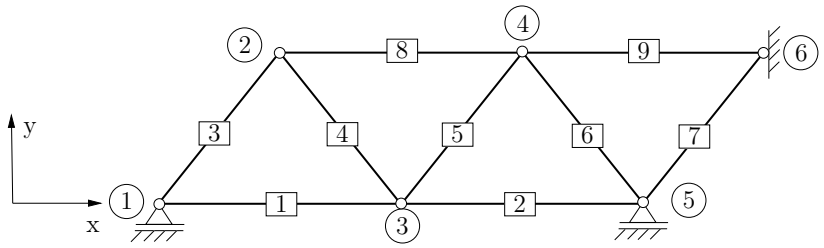


Abbildung 5.4

Die `connectivity`-Matrix lautet:

$$\texttt{connectivity} = \begin{bmatrix} 1 & 3 & 1 & 2 & 3 & 4 & 5 & 2 & 4 \\ 3 & 5 & 2 & 3 & 4 & 5 & 6 & 4 & 6 \end{bmatrix}$$

Das Randwertproblem besteht aus neun Stabelementen, sechs Knoten und vier essentiellen Randbedingungen, d.h.

$$\texttt{num\_ele} = 9, \quad \texttt{num\_nodes} = 6, \quad \texttt{num\_bc} = 4.$$

Die Elementsteifigkeitsmatrix $\boldsymbol{k}^e$ hat die Dimension $[\mathtt{k}^{\mathtt{e}}_{\mathtt{dim}} \times \mathtt{k}^{\mathtt{e}}_{\mathtt{dim}}]$ mit

$$\mathtt{k}^{\mathtt{e}}_{\mathtt{dim}} = \texttt{nodes\_ele} * \texttt{dof\_nodes} = 4.$$

Die Dimension der globalen Steifigkeitsmatrix $\underline{\boldsymbol{K}}$ unter Berücksichtigung der Dirichlet Randbedingungen ist $[\texttt{num\_eq} \times \texttt{num\_eq}]$ mit $\texttt{num\_eq} = 8$ (5.1). Die `assembleid`-Matrix lautet

$$\texttt{assembleid} = \begin{bmatrix} 1 & 2 & 4 & 6 & 8 & 0 \\ 0 & 3 & 5 & 7 & 0 & 0 \end{bmatrix}.$$

Mit ihrer Hilfe werden die einzelnen Einträge $[\boldsymbol{k}^e]_{\mathrm{ij}}$ der Elementsteifigkeitsmatrix den Einträgen $[\boldsymbol{K}]_{\mathrm{IJ}}$ der globalen Steifigkeitsmatrix zugeordnet. Bevor dies geschieht wird jedoch zunächst die leere globale Steifigkeitsmatrix aufgestellt mit

$$\mathrm{K_{IJ}} = 0.0 \quad \text{für} \quad \mathrm{I,J} = 1, ...8 \, .$$

Nun wird der Element-Schleifendurchlauf gestartet. In jedem Iterationsschrit wird zunächst die lokale Elementsteifigkeitsmatrix $\underline{\boldsymbol{k}}^e$ mit `PlaneTrussElementStiffness.m` erzeugt. Daraufhin wird mithilfe der Subroutine `PlaneTrussAssemble.m` die globale Steifigkeitsmatrix $\underline{\boldsymbol{K}}$ geupdatet. Um diesen Vorgang zu verdeutlichen wird der erste Iterationsschritt (Element $e = 1$) betrachtet. Die Positionen I und J in der globalen Matrix können den Einträgen der `connectivity`- und `assembleid`- Matrix entnommen werden:

$$\mathrm{I} = \texttt{assembleid}(:, \texttt{connectivity}(1,1)) = [\ \texttt{1}\ \ \texttt{0}\ ]$$
$$\mathrm{J} = \texttt{assembleid}(:, \texttt{connectivity}(1,2)) = [\ \texttt{4}\ \ \texttt{5}\ ]$$

mit der Elementsteifigkeitsmatrix

$$\underline{\boldsymbol{k}}^{\boxed{1}} = \begin{array}{c|cccc|l} & 1 & 0 & 4 & 5 & \text{global dof} \\ \hline & \mathrm{k}^{\boxed{1}}_{11} & \mathrm{k}^{\boxed{1}}_{12} & \mathrm{k}^{\boxed{1}}_{13} & \mathrm{k}^{\boxed{1}}_{14} & 1 \\ & \mathrm{k}^{\boxed{1}}_{21} & \mathrm{k}^{\boxed{1}}_{22} & \mathrm{k}^{\boxed{1}}_{23} & \mathrm{k}^{\boxed{1}}_{24} & 0 \\ & \mathrm{k}^{\boxed{1}}_{31} & \mathrm{k}^{\boxed{1}}_{32} & \mathrm{k}^{\boxed{1}}_{33} & \mathrm{k}^{\boxed{1}}_{34} & 4 \\ & \mathrm{k}^{\boxed{1}}_{41} & \mathrm{k}^{\boxed{1}}_{42} & \mathrm{k}^{\boxed{1}}_{43} & \mathrm{k}^{\boxed{1}}_{44} & 5 \end{array} \, .$$

Die ersten zwei Zeilen und Spalten von $\underline{\mathrm{k}}^{\boxed{1}}$ sind dem vertikalen und horizontalem Freiheitsgrad des globalen Knotens 1 zugeornet; die dritte und vierte Zeile und Spalte sind dem globalen Knoten 3 zugeordnet. Mit der dieser Information folgt nun der Update-Vorgang

$$\begin{aligned} &\mathrm{K_{11}} \Longleftarrow K_{11} + k^{\boxed{1}}_{11} \, , \quad \mathrm{K_{14}} \Longleftarrow K_{14} + k^{\boxed{1}}_{13} \, , \quad \mathrm{K_{15}} \Longleftarrow K_{15} + k^{\boxed{1}}_{14} \, , \\ &\mathrm{K_{41}} \Longleftarrow K_{41} + k^{\boxed{1}}_{31} \, , \quad \mathrm{K_{44}} \Longleftarrow K_{44} + k^{\boxed{1}}_{33} \, , \quad \mathrm{K_{45}} \Longleftarrow K_{45} + k^{\boxed{1}}_{43} \, , \\ &\mathrm{K_{51}} \Longleftarrow K_{51} + k^{\boxed{1}}_{41} \, , \quad \mathrm{K_{54}} \Longleftarrow K_{54} + k^{\boxed{1}}_{34} \, , \quad \mathrm{K_{55}} \Longleftarrow K_{55} + k^{\boxed{1}}_{44} \, . \end{aligned} \tag{5.2}$$

Es wird deutlich, dass die Einträge der zweiten Zeile und Spalte nicht in das Update der globalen Matrix eingehen, da die zugehörigen globalen Freiheitsgrade aufgrund der Dirichlet-Randbedingungen in der `assembleid`-Matrix mit Null gekennzeichnet sind. Für den zweiten Iterationsschritt $e = 2$, bei dem das zweite Element betrachtet wird, folgt:

$$\underline{\boldsymbol{k}}^{[2]} = \begin{array}{c|cccc|c} & 4 & 5 & 8 & 0 & \text{global dof} \\ \hline & \mathrm{k}_{11}^{[2]} & \mathrm{k}_{12}^{[2]} & \mathrm{k}_{13}^{[2]} & \mathrm{k}_{14}^{[2]} & 4 \\ & \mathrm{k}_{21}^{[1]} & \mathrm{k}_{22}^{[2]} & \mathrm{k}_{23}^{[2]} & \mathrm{k}_{24}^{[2]} & 5 \\ & \mathrm{k}_{31}^{[2]} & \mathrm{k}_{32}^{[2]} & \mathrm{k}_{33}^{[2]} & \mathrm{k}_{34}^{[1]} & 8 \\ & \mathrm{k}_{41}^{[2]} & \mathrm{k}_{42}^{[2]} & \mathrm{k}_{43}^{[2]} & \mathrm{k}_{44}^{[1]} & 0 \end{array} .$$

$$\begin{aligned} &\mathrm{K}_{44} \Longleftarrow K_{44} + k_{11}^{[2]} , \quad \mathrm{K}_{45} \Longleftarrow K_{45} + k_{12}^{[2]} , \quad \mathrm{K}_{48} \Longleftarrow K_{48} + k_{13}^{[2]} , \\ &\mathrm{K}_{54} \Longleftarrow K_{54} + k_{21}^{[2]} , \quad \mathrm{K}_{55} \Longleftarrow K_{55} + k_{22}^{[2]} , \quad \mathrm{K}_{58} \Longleftarrow K_{58} + k_{23}^{[2]} , \\ &\mathrm{K}_{84} \Longleftarrow K_{84} + k_{31}^{[2]} , \quad \mathrm{K}_{85} \Longleftarrow K_{85} + k_{32}^{[2]} , \quad \mathrm{K}_{88} \Longleftarrow K_{88} + k_{33}^{[2]} . \end{aligned} \tag{5.3}$$

In gleicher Weise wird für die weiteren Elemente verfahren bis der Schleifendurchlauf über alle Elemente abgeschlossen ist. Schließlich bleibt noch zu berücksichtigen, dass ein Element je nach vorliegenden Dirichlet Randbedingungen eine unterschiedliche Anzahl von Freiheitsgraden aufweisen kann. Um diese Tatsache algorithmisch zu handhaben kann die Subroutine `PlaneTrussAssemble.m` als Kontrollroutine betrachtet werden welche mithilfe von Fallunterscheidung prüft wieviele aktive Freiheitsgrade im jeweiligen Element vorliegen. Besitzt ein Element nur einen aktiven Freiheitsgrad so wird zum Update der globalen Steifigkeitsmatrix die Unterfunktion `Assemble1.m` aufgerufen. Liegen zwei aktive Freiheitsgrade vor wird die Unterfunktion `Assemble2.m` aufgerufen, und so weiter. In den Unterfunktionen selbst wird dann das Update, wie in (5.2) und (5.3) veranschaulicht, durchgeführt

c) Versehen Sie die Hauptdatei `PlaneTruss.m` nun mit einem Interface, welches den globalen Lastvektor (Neumann-Randbedingungen) einliest.

d) Lassen Sie das Programm schließlich das globale Gleichungssystem lösen und die `connectivity`-Matrix, $\underline{\boldsymbol{K}}$, $\underline{\boldsymbol{R}}$ und $\underline{\boldsymbol{D}}$ ausgeben.

Hinweis: Zum Verdeutlichen des Einlesevorgangs des Lastvektors `R` ([`num_eq` $\times$ `1`]) wird wieder das Beispiel aus Abbildung 5.3 betrachtet. Wie Abbildung 5.3 zu entnehmen ist, besteht dieser aus dem lokalen Vektor $\boldsymbol{F}^{②}$, wofür folgende Werte gegeben seien:

$$\mathrm{F}^{②} = [\mathrm{F}_1^{②}, \mathrm{F}_2^{②}]^{\mathrm{T}} = [0.01\,\mathrm{kN}, -0.05\,\mathrm{kN}]^{\mathrm{T}}.$$

Das Interface sieht folgendermaßen aus:

```
give the total number of nodes with non-zero load vectors
                                    num_loads = 1
            input of load vectors for i from 1 to num_loads
  no. of global node with non-zero load vector = 2
                  force in global x-direction = 0.01
                  force in global y-direction = -0.05
```

Der erste Output ist die zur `connectivity`-Matrix gehörige Tabelle

```
============================
|  e  |  Node i  |  Node j  |
============================
|  1  |     1    |     2    |
|  2  |     2    |     3    |
============================
```

Da $\underline{\boldsymbol{K}}$, $\underline{\boldsymbol{R}}$ und $\underline{\boldsymbol{D}}$ in dem Programm als Variablen auftreten kann der Output dieser Größen nach der Lösung direkt erfolgen.

e) Verifizieren, ob ihr Programm richtig funktioniert, indem Sie Ihr FE-Programm für das Fachwerk aus Aufgabe 5.1: auswerten und mit der Lösung von Aufgabe 5.1:c) vergleichen.

f) Was passiert wenn Sie bei der Eingabe keine Lagerungsrandbedingungen angeben? Wie lässt sich das beobachtete Verhalten begründen?

Übung 6: Einführung in FEAPpv

Die kommenden Übungen werden sich mit der Implementierung von Elementen innerhalb der Finite Elemente Software FEAPpv befassen. FEAPpv ist für die Nutzung auf Unix-/ Linux-Systemen konzipiert, die Software Cygwin ermöglicht jedoch auch die Nutzung unter Windows. Die Installation innerhalb der Cygwin-Umgebung ist nachfolgend beschrieben.

Installation von FEAPpv unter Cygwin

Zur erstmaligen Installation von FEAPpv im CIP-Pool müssen folgende Schritte durchgeführt werden:

1.) ein Cygwin64-Terminal öffnen,

2.) Homeverzeichnis von Cygwin unter C:/cygwin64/home/<RUB-login> öffnen und am Ende der Datei `.bashrc` die folgenden Zeilen einfügen:

```
export FEAPPVHOME4_1=/cygdrive/z/feappv41/ver41
alias feappv='$FEAPPVHOME4_1/main/feappv.exe',
alias gnuplot="/cygdrive/c/'Program Files'/blueCFD-Core-2017/msys64/...
...mingw64/bin/gnuplot.exe",
```

anschließend die Datei zur späteren Nutzung auf das eigene Laufwerk Z sichern,

3.) den bereitgestellten Ordner `feappv41` im persönlichen Laufwerk Z ablegen,

4.) unter alle Programme/Cygwin-X auf „XWin Server“ klicken und im Anschluss durch Rechtsklick auf das Cygwin-Symbol in der Taskleiste unter Systemwerkzeuge ein Cygwin-Terminal starten,

5.) mit dem Befehl `cd /cygdrive/z/feappv41/ver41` in den Ordner `feappv41/ver41` navigieren und das Programm mit dem Befehl `make install` kompilieren,

6.) zum Testen des Programms in den Ordner `calc/0_beispiel` wechseln, den Befehl `feappv` eingeben und den weiteren Anweisungen folgen.

Wenn Änderungen am Programm vorgenommen werden, muss dieses neu kompiliert werden. Das erfolgt durch Eingabe von `make` im Ordner `ver41`. Einige häufig verwendete Konsolen-Befehle sind auf der nächsten Seite zusammengefasst.

Einige nützliche Konsolenbefehle

Befehl	Auswirkung
ls	Auflisten des Inhalts des aktuellen Verzeichnisses
cd <Pfad>	aus dem aktuellen Verzeichnis nach ¡Pfad¿ navigieren
cd ..	eine Verzeichnisebene nach oben wechseln
cd /cygdrive/z	ins Laufwerk Z navigieren
clear	Konsoleninhalt löschen
exit	Konsole schließen

Kurzübersicht FEAP-Kommandos

`tang,,1`	start one iteration step
`disp,,`*num*	print coordinates and displacements of node *num*
`plot`	enter the plot-command-environment
`plot,mesh`	show mesh
`plot,node,`*num*	show number of node *num* *num*=blank, plot all numbers)
`plot,elem,`*num*	show number of element *num* (*num*=blank, plot all numbers)
`plot,defo,`*factor*`,1`	switch to deformed configuration with scaling by *factor*
`plot,unde`	switch to undeformed configuration
`plot,boun`	show boundary conditions
`plot,load`	show loads
`plot,post`	start/end plot to postscript-file
`plot,stre,`*index*`,,`*flag*	plot stress distribution, *index*: 1→ σ_{11}, 2→ σ_{22}, ... ; *flag*: 0=mesh, 1=no mesh
`plot,wipe`	clear plot window

Hinweis: Weitere Befehle und Erläuterungen lassen sich dem in FEAPpv enthaltenden Benutzerhandbuch unter dem Dateipfad `ver41/manual/manual41.pdf` entnehmen.

Hinweis zum Erstellen von Plots mithilfe von `gnuplot`

`Gnuplot` ist ein skript- bzw. kommandozeilengesteuerte Computerprogramm zur grafischen Darstellung von Daten. Entsprechende Skriptdateien werden mit folgendem Befehl ausgeführt:

```
gnuplot <scriptname>.gnu
```

Aufgabe 6.1: Cook's Membran Problem

Eine FEAPpv-kompatible Input-Datei für das dargestellte Problem soll erstellt werden. Nutzen sie die in dieser Übung bereitgestellte Beispieldatei `I_test` als Vorlage und erzeugen sie die FE-Lösung für die folgenden Netzverfeinerungsstufen:

	Elemente in x-Richtung (nx)	Elemente in y-Richtung (ny)
1)	10	5
2)	20	10
3)	40	20
4)	80	40
5)	160	80

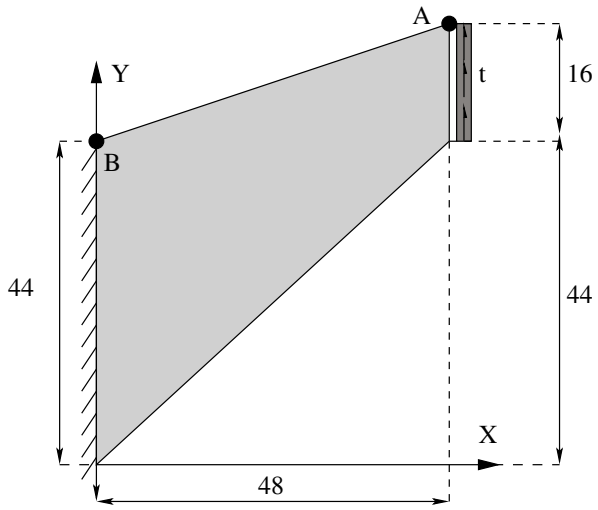


Abbildung 6.1: Cook's Membran Geometrie

Materialparameter: $E = 210000\,\text{MPa}$, $\nu = 0.3$
Neumann-Randbedingung: $t = 5000\,\text{MPa}$

a) Berechnen Sie die Verschiebung in Vertikalrichtung im Punkt `A` für die verschiedenen Netzverfeinerungsstufen 1)-5) und Veranschaulichen Sie die Ergebnisse in einem entsprechenden Diagramm.

Hinweis: Speichern Sie dazu die jeweiligen Berechnungsergebnisse in der Datei `u-nelem.dat` und nutzen Sie das beigefügte Gnuplot-Skript `u-nelem.gnu`.

b) Erzeugen Sie die folgenden Plots

- Netz im Ausgangszustand mit Dirichiet- und Neumann Randbedingungen basierend auf Netzverfeinerungsstufe 2),
- Geometrie im Verformten Zustand mit den Spannungs-Contourplots $\sigma_{11}, \sigma_{22}, \sigma_{33}$ basierend auf Netzverfeinerungsstufe 5).

Übung 7: Lineares Dreieckselement

Ziel dieser Übung ist es die benutzereigene FEAPpv Elementsubroutine `elmt03` für lineare Dreieckselemente zu erstellen und in das Programm einzubinden. Dazu werden in den Teilaufgaben stückweise die Unter-Subroutinen erstellt, mithilfe derer schließlich die Elementsteifigkeitsmatrix des linearen Verschiebungs-Dreieckselementes erzeugt wird.

Kompilieren und Ausführen von Fortran Programmen

Zum Kompilieren und Ausführen der Beispielhaften Fortran Datei `hello_world.f` werden auf dem CIP-Pool Rechner folgende Schritte durchgeführt

1.) den bereitgestellten Ordner `ue07_f` im persönlichen Laufwerk Z ablegen

2.) unter alle Programme/Cygwin-X auf „XWin Server“ klicken und im Anschluss durch Rechtsklick auf das Cygwin-Symbol in der Taskleiste unter Systemwerkzeuge ein Cygwin-Terminal starten,

3.) mit dem Befehl `cd /cygdrive/z/ue07_f` in den Ordner `ue7_f` navigieren

4.) in der Konsole `gfortran -o hello_world hello_world.f` eingeben. Mit diesem Aufruf wird dem Compiler `gfortran` der Befehl gegeben die ausführbare Datei `hello_world` aus dem Fortran-Quellcode der Datei `hello_world.f` zu kompilieren.

5.) in der Konsole `./hello_world` eingeben um das Progamm auszuführen.

(ggf. muss bei Windows-Systemen bei den Kommandozeileneingaben die Endung `.exe` bei `gfortran` und `hello_world` hinzugefügt werden.)

Schritte 4 und 5 lassen sich auf jeden Fortran Quellcode `<filename>.f` anwenden.

Hinweise zum Programmieren mit Fortran

Zur Programmierung können nur die Spalten $6 - 72$ verwendet werden.

Befehl	**Wirkung**

`programm <program name>`
...
...
`end program`
	erste und letzte Zeile eines Programms

`integer i`
`real*8 a`
`real*8 b(10)`
	Initialisierung von der Variablen i als Integer, a als reelle Zahl und b als Array mit 10 reellen Einträgen

`do i=1,100`
...
...
`end do`
	Schleife, in der die Variable i von 1 bis 100 iteriert wird.

`if(r.lt.0)then`
... `Block A` ...
`else`
... `Block B`...
`end if`
	„wenn, dann“-Abfrage; hier: wenn r kleiner als 0 ist, wird Block A ausgeführt, ansonsten Block B

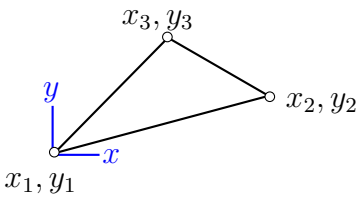
Aufgabe 7.1: Berechnen der Elastizitätsmatrix

Zur Berechnung der Spannungskomponenten in der x-y-Ebene lautet die Elastizitätsmatrix $\underline{\mathbb{C}}^{(V)}$ in Voigt-Notation:

$$\underline{\mathbb{C}} = \begin{bmatrix} \lambda + 2\,\mu & \lambda & 0 \\ \lambda & \lambda + 2\,\mu & 0 \\ 0 & 0 & \mu \end{bmatrix} \tag{7.1}$$

Vervollständigen Sie in der Datei `elmt03.f` die Subroutine `dmat03`, sodass mithilfe der Eingabeparameter `yo` (E-Modul) und `nu` (Querkontraktionszahl) die einzelnen Komponenten der Elastizitätsmatrix `aa` berechnet werden.

Hinweis: Um die in dieser und in den folgenden Aufgaben erzeugten Subroutinen zu testen kann die Datei `t1_test.f` verwendet werden. Diese beinhaltet einen Parametersatz (`t1_param`) für ein beispielhaftes Dreieckselement (vgl. Abbildung 7.1) und gibt die mit den Subroutinen berechneten Ergebnisse in der Konsole aus (`t1_output_formats`)



E	10000 MPa
ν	0.3
x_1, y_1	0.0, 0.0 mm
x_2, y_2	1.0, 0.3 mm
x_3, y_3	0.5, 0.8 mm

Abbildung 7.1: Knotenkoordinaten und Elastizitätsparameter eines Beispielelementes

Aufgabe 7.2: Berechnen der B-Matrix

Die sogennante B-Matrix $\underline{\boldsymbol{B}}^{\mathrm{e}}$ hat für das lineare Dreieckselement die Form

$$\underline{\boldsymbol{B}}^{\mathrm{e}} = \begin{bmatrix} N_{1,x} & 0 & N_{2,x} & 0 & N_{3,x} & 0 \\ 0 & N_{1,y} & 0 & N_{2,y} & 0 & N_{3,y} \\ N_{1,y} & N_{1,x} & N_{2,y} & N_{2,x} & N_{3,y} & N_{3,x} \end{bmatrix}, \tag{7.2}$$

wobei die Ableitungen der Ansatzfunktionen N_I für $I \in \{1, 2, 3\}$ nach den physikalischen Koordinaten mit folgender Formel aufgestellt werden:

$$\begin{bmatrix} N_{1,x} \\ N_{2,x} \\ N_{3,x} \end{bmatrix} = \frac{1}{2A^{\mathrm{e}}} \begin{bmatrix} y_2 - y_3 \\ y_3 - y_1 \\ y_1 - y_2 \end{bmatrix}, \qquad \begin{bmatrix} N_{1,y} \\ N_{2,y} \\ N_{3,y} \end{bmatrix} = \frac{1}{2A^{\mathrm{e}}} \begin{bmatrix} x_3 - x_2 \\ x_1 - x_3 \\ x_2 - x_1 \end{bmatrix} \tag{7.3}$$

Dabei stellt A^{e} die Elementfläche dar, welche sich mithilfe der Determinante der Transformationsmatrix $\underline{\boldsymbol{A}}^{\mathrm{e}}$ berechnen lässt:

$$2A^{\mathrm{e}} = \det \underline{\boldsymbol{A}}^{\mathrm{e}} = (x_2 - x_1)(y_3 - y_1) + (x_3 - x_1)(y_1 - y_2) \quad \text{mit} \quad \underline{\boldsymbol{A}}^{\mathrm{e}} = \begin{bmatrix} 1 & 1 & 1 \\ x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \end{bmatrix} \tag{7.4}$$

Vervollständigen Sie in der Datei `elmt03.f` die Subroutine `bmat03`, sodass mithilfe der Eingabeparameter `det` (Elementfläche $2A^{\mathrm{e}}$) und `xl` (Knotenkoordinatenmatrix $[2\times3]$) die einzelnen Komponenten der B-Matrix `bmat` berechnet werden.

Aufgabe 7.3: Elementsteifigkeitsmatrix aufstellen

Für lineare Dreieckselemente lässt sich die Elementsteifigkeitsmatrix $\underline{\boldsymbol{k}}^{\mathrm{e}}$ wie folgt berechnen:

$$\begin{aligned} \underline{\boldsymbol{k}}^{e} &= \int_{\mathcal{B}^{\mathrm{e}}} (\underline{\boldsymbol{B}}^{\mathrm{e}})^T \underline{\mathbb{C}} \underline{\boldsymbol{B}}^{\mathrm{e}} \, \mathrm{d}V = A^{\mathrm{e}} (\underline{\boldsymbol{B}}^{\mathrm{e}})^T \underline{\mathbb{C}} \underline{\boldsymbol{B}}^{\mathrm{e}} \\ [6 \times 6] &= [6 \times 3][3 \times 3][3 \times 6] \end{aligned} \tag{7.5}$$

Hierzu sei eine Scheibendicke von h=1 mm angenommen. Vervollständigen Sie in der Datei `elmt03.f` die Subroutine `kemat03`, sodass mithilfe der Eingabeparameter `bmat` (B-Matrix), `aa` (Elastizitätsmatrix), `det` ($2A^{\mathrm{e}}$) und `nst` (Dimension von $\underline{\boldsymbol{k}}^{\mathrm{e}}$) die einzelnen Komponenten der Elementsteifigkeitsmatrix `s` berechnet werden.

Hinweis: Anders als bei MATLAB (vgl. Übung 5) werden Matrix-Multiplikationen in Fortran nicht automatisch durchgeführt. Die einzelnen Komponenten des resultierenden Matrixproduktes müssen per Schleifendurchlauf berechnet werden. Dazu bietet sich an die jeweiligen Matrixmultiplikationen schrittweise zu berechnen und zunächst das Zwischenprodukt

$$\texttt{btc} := (\underline{\boldsymbol{B}}^{\mathrm{e}})^T \underline{\mathbb{C}}$$

auszurechnen, mithilfe dessen daraufhin

$$\texttt{s} := A^{\mathrm{e}} (\texttt{btc}) \underline{\boldsymbol{B}}^{\mathrm{e}}$$

berechnet werden kann.

Aufgabe 7.4: Einbinden in FEAPpv und Beispielrechnung

a) Vervollständigen Sie die Elementsubroutine `elmt03` indem sie die aus den voherigen Teilaufgaben aufgestellten Unter-Subroutinen nacheinander aufrufen um die Elementsteifigkeitsmatrix aufzustellen.

b) Ersetzen Sie die nun vervollständigte Datei `elmt03.f` in dem Dateipfad `$FEAPPVHOME4_1/user` und updaten Sie das FEAPpv-Hauptprogram mit dem Befehl `make`.

(Eventuell ist es notwendig die in dem Verzeichnis befindliche Datei `elmt03.f` dem `$FEAPPVHOME4_1/user`-Dateipfad einmal zu öffnen und zu speichern.)

c) Führen die Cook's Membrane-Problem Rechnung (Input-Datei: `I_cm`) für einen der Netzverfeinerungzustände aus Aufgabe 6.1a) mit dem soeben erstellten User-Element durch und vergleichen Sie die Lösungen.

Hinweis: Benutzerdefinierte Elemente werden in der FEAPpv Inputdatei durch den Befehl

```
mate <number>
  user, <user elmt number>
  <user mat par1>  <user mat par2> ...
```

aufgerufen. In dem hier vorliegenden Fall sollten die Ergebnisse des implementierten User-Elementes

```
mate 1
  user, 3
  E nu
```

mit denen des in der vorherigen Aufgabe verwendeten programmeigenen linearen Dreieckselementes

```
mate 1
  soli
  elas isot E nu
```

mit den Initialisierungsgrößen $\{\texttt{ndm}, \texttt{nen}\} = \{2, 3\}$ übereinstimmen.

Übung 8: Ansatzfunktionen

Aufgabe 8.1: Verständnisfragen

a) Wie lassen sich grundsätzlich FE-Elementansatzfunktionen bestimmen?

b) Welche Eigenschaft der Ansatzfunktionen wird dabei genutzt?

c) Was ist die Idee des isoparametrischen Konzeptes?

d) Welche Vorteile hat es die FE-Approximation über das Referenzelement im Parameterraum durchuführen?

Aufgabe 8.2: Isoparametrisches Viereckselement

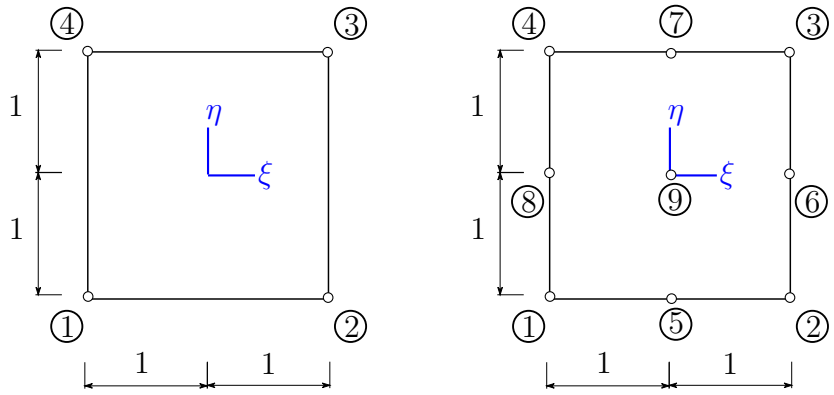


Abbildung 8.1: lineares und quadratisches Referenzviereckselement im Parameterraum

a) Bestimmen Sie die Ansatzfunktionen N_1 und N_2 für das lineare Viereckselement aus Abbildung 8.1

b) Bestimmen Sie die Ansatzfunktionen N_2 und N_6 für das quadratische Viereckselement aus Abbildung 8.1

Aufgabe 8.3: Referenzdreieckselement

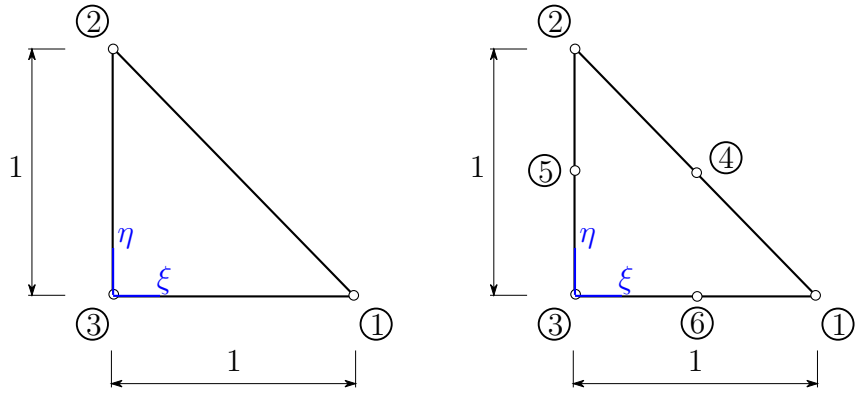


Abbildung 8.2: lineares und quadratisches Referenzdreieckselement im Parameterraum

a) Bestimmen Sie die Ansatzfunktionen für das lineare Dreieckselement aus Abbildung 8.2 mithilfe des allgemeinen Ansatzes

b) Wie lauten die Ansatzfunktionen N_1 und N_6 für das quadratische Dreieckselement aus Abbildung 8.2. ?

c) Wie lauten die Ansatzfunktionen für ein dreidimensionales lineares Tetraederelement?

Übung 9: Quadratisches Dreieckselement

In dieser Übung wird die FEAPpv Elementsubroutine `elmt04` für isoparametrische lineare und quadratische Dreieckselemente vervollständigt. Ziel ist es insbesondere die Einträge der B-matrix $(N_{I,x}, N_{I,y})$ zu berechnen mithilfe der kinematischen Beziehungen zwischen Parameter- und physikalischen Koordinaten (Jacobimatrix $\underline{\boldsymbol{J}}$, Jacobi-Determinante $\det \underline{\boldsymbol{J}}$ und Inverse der Jacobimatrix $\underline{\boldsymbol{J}}^{-1}$).

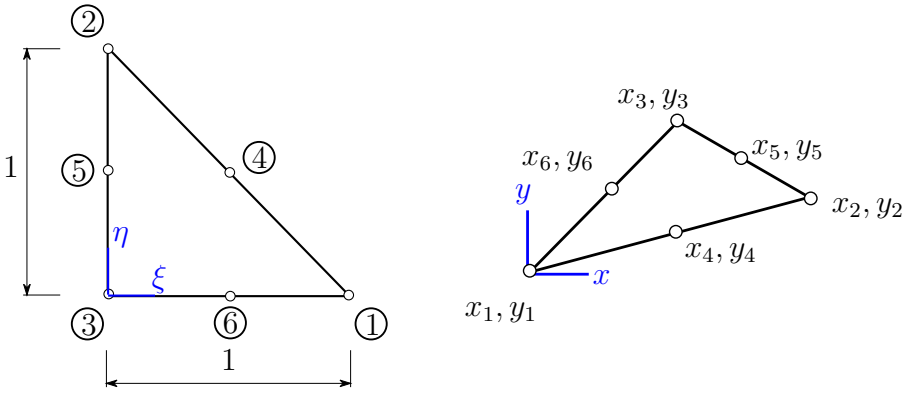


Abbildung 9.1: Quadratisches Referenzdreieckselement im Parameterraum und Beispielement im physikalischen Raum

Aufgabe 9.1: Ableitung der Ansatzfunktionen

Die Ansatzfunktionen des quadratischen Dreieckselementes lauten

$$\begin{bmatrix} N_1 \\ N_2 \\ N_3 \\ N_4 \\ N_5 \\ N_6 \end{bmatrix} = \begin{bmatrix} \lambda_1(2\lambda_1 - 1) \\ \lambda_2(2\lambda_2 - 1) \\ \lambda_3(2\lambda_3 - 1) \\ 4\lambda_1\lambda_2 \\ 4\lambda_2\lambda_3 \\ 4\lambda_3\lambda_1 \end{bmatrix}, \quad \text{wobei sich die Flächenkoordinaten mit} \quad \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \end{bmatrix} = \begin{bmatrix} \xi \\ \eta \\ 1 - \xi - \eta \end{bmatrix} \tag{9.1}$$

durch die Parameterkoordinaten ξ und η ausdrücken lassen. Mithilfe der Ableitungen der Ansatzfunktionen nach den Parameterkoordinaten und den physikalischen Knotenkoordinaten lässt sich mithilfe des isoparametrischen Konzeptes die Jacobimatrix

$$\underline{\boldsymbol{J}} = \sum_{I=1}^{\text{nen}} \begin{bmatrix} x_I N_{I,\xi} & x_I N_{I,\eta} \\ y_I N_{I,\xi} & y_I N_{I,\eta} \end{bmatrix} = \begin{bmatrix} x_1 & x_2 & \dots & x_{\text{nen}} \\ y_1 & y_2 & \dots & y_{\text{nen}} \end{bmatrix} \begin{bmatrix} N_{1,\xi} & N_{1,\eta} \\ N_{2,\xi} & N_{2,\eta} \\ \vdots & \vdots \\ N_{\text{nen},\xi} & N_{\text{nen},\eta} \end{bmatrix} \tag{9.2}$$

aufstellen. In der Datei `elmt04.f` in der Subroutine `shape04` wird die Koordinatenmatrix mit `xl` und die Matrix bestehend aus Ableitungen der Ansatzfunktionen nach Parameterkoordinaten mit `sh0` bezeichnet. (9.2) wird demzufolge berechnet durch $\texttt{xl}.\texttt{sh0}^T$.

a) Implementieren Sie (9.2) zum Aufstellen der Jacobimatrix in der Subroutine `shape04`.

Zur Berechnung der Ableitung der Ansatzfunktionen nach den physikalischen Koordinaten wird die Inverse der Jacobimatrix benötigt. Diese lässt sich im 2D-Fall direkt berechnen mit:

$$\underline{\boldsymbol{J}}^{-1} = \frac{1}{\det \underline{\boldsymbol{J}}} \begin{bmatrix} J_{22} & -J_{12} \\ -J_{21} & J_{11} \end{bmatrix} \tag{9.3}$$

b) Ergänzen Sie `shape04` um die Jacobideterminante und die Inverse der Jacobimatrix.

Die Ableitungen der Ansatzfunktionen lassen sich mit

$$\begin{bmatrix} N_{I,x} \\ N_{I,y} \end{bmatrix} = \underline{\boldsymbol{J}}^{-T} \begin{bmatrix} N_{I,\xi} \\ N_{I,\eta} \end{bmatrix} \tag{9.4}$$

durch das folgende Matrixprodukt aufstellen:

$$\begin{bmatrix} N_{1,x} & \dots & N_{\text{nen},x} \\ N_{1,y} & \dots & N_{\text{nen},y} \end{bmatrix} := \begin{bmatrix} \texttt{shp(1,*)} \\ \texttt{shp(2,*)} \end{bmatrix} = \underline{\boldsymbol{J}}^{-T}.\texttt{sh0} \tag{9.5}$$

c) Vervollständigen Sie `shape04` durch die Berechnung der Ableitungen (9.5)

Hinweis: Wie in Übung 7 kann die Datei `t2_test.f` verwendet werden um die Datei `elmt04.f` zu testen. Der zur Tabelle 9.2 gehörige Parametersatz für ein entsprechendes Beispielelement befindet sich in der Datei `t2_param`.

E	10000 MPa		
ν	0.3		
x_1, y_1	$0.0, 0.0$ mm	x_4, y_4	$0.50, 0.15$ mm
x_2, y_2	$1.0, 0.3$ mm	x_5, y_5	$0.75, 0.55$ mm
x_3, y_3	$0.5, 0.8$ mm	x_6, y_6	$0.25, 0.40$ mm

Tabelle 9.2: Parameter des quadratischen Dreiecks-Beispielelementes

Aufgabe 9.2: Konvergenzstudie

a) Ersetzen Sie die vervollständigte Datei `elmt04.f` in dem Dateipfad `$FEAPPVHOME4_1/user` und updaten Sie das FEAPpv-Hauptprogram mit dem Befehl `make` (vgl Aufgabe 7.4).

b) Führen Sie eine Beispielrechnung mit dem linearen und quadratischen User-Element `elmt04.f` durch und vergleichen sie die Lösungen mit denen der entsprechenden programmeigenenen Dreieckselemente.

c) Führen Sie die Netzkonvergenzstudie aus Aufgabe 6.1a) für das quadratische Dreieckselement durch und vergleichen Sie die Ergebnisse mit denen des linearen Dreieckselementes. Welcher der beiden Elementtypen hat hier ein vorteilhafteres Konvergenzverhalten?

Übung 10: Vierknotenelement

In dieser Übung wird die FEAPpv Elementsubroutine `elmt04` aus den vorherigen Aufgaben um das isoparametrische Vierknotenelement (vgl. Abbildung 10.1) erweitert.

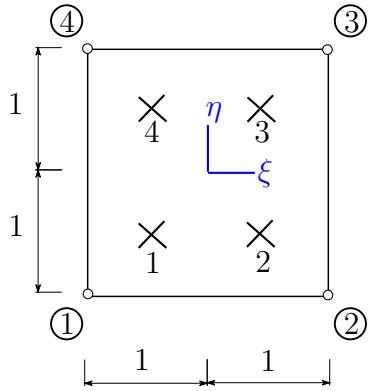


Abbildung 10.1: lineares Referenzviereckselement mit Gausspunkten

Grundlagen: Mithilfe der Gausspunkte wird das Integral der Elementsteifigkeitsmatrix $\underline{\boldsymbol{k}}^{\mathrm{e}}$ wie folgt numerisch berechnet:

$$\underline{\boldsymbol{k}}^{\mathrm{e}} = \int_{\mathcal{B}^{\mathrm{e}}} (\underline{\boldsymbol{B}}^{\mathrm{e}})^T \underline{\mathbb{C}}\, \underline{\boldsymbol{B}}^{\mathrm{e}}\, \mathrm{d}V = \sum_{l=1}^{n_{\mathrm{GP}}=4} (\underline{\boldsymbol{B}}^{\mathrm{e}}(\boldsymbol{\xi}_l))^T \underline{\mathbb{C}}\, \underline{\boldsymbol{B}}^{\mathrm{e}}(\boldsymbol{\xi}_l) \det \underline{\boldsymbol{J}}(\boldsymbol{\xi}_l) \mathrm{w}_l \tag{10.1}$$

Dabei sind $\boldsymbol{\xi}_l = [\xi_l, \eta_l]^T$ die Ortsvektoren der Gausspunkte im Parameterraum und w_l die zugehörigen Wichtungsfaktoren. Für das Vierknotenelement sind diese Tabelle 10.3 zu entnehmen.

i	ξ_l	η_l	w_l
1	$-1/\sqrt{3}$	$-1/\sqrt{3}$	1
2	$1/\sqrt{3}$	$-1/\sqrt{3}$	1
3	$1/\sqrt{3}$	$1/\sqrt{3}$	1
4	$-1/\sqrt{3}$	$1/\sqrt{3}$	1

Tabelle 10.3: Gausspunktkoordinaten und Wichtungsfaktoren des Vierknotenelementes

Die Summe in (10.1) wird im FE-Programm üblicherweise im Schleifendurchlauf aufgestellt, wobei $\underline{\boldsymbol{k}}^{\mathrm{e}}$ in jedem Iterationsschritt l geupdated wird:

$\underline{\boldsymbol{k}}^{\mathrm{e}} = \underline{\boldsymbol{k}}^{\mathrm{e}}_0$ (in linear elasticity $= \mathbf{0}$)
do $l = 1$, n_{GP}
  $\underline{\boldsymbol{k}}^{\mathrm{e}} \Longleftarrow \underline{\boldsymbol{k}}^{\mathrm{e}} + (\underline{\boldsymbol{B}}^{\mathrm{e}}(\boldsymbol{\xi}_l))^T \underline{\mathbb{C}}\, \underline{\boldsymbol{B}}^{\mathrm{e}}(\boldsymbol{\xi}_l) \det \underline{\boldsymbol{J}}(\boldsymbol{\xi}_l) \mathrm{w}_l$
end do

Desweiteren werden im Postprocessing, nachdem das FE-Problem gelöst ist und der Elementverschiebungsvektor $\underline{\boldsymbol{d}}^{\mathrm{e}}$ bekannt ist, an den Gausspunkten die Spannungen und Verzerrungen ausgewertet. Mithilfe der aus der Vorlesung bekannten Element-Interpolationsfunkionen $\underline{\boldsymbol{N}}^{\mathrm{e}}$ und $\underline{\boldsymbol{B}}^{\mathrm{e}}$ kann der Gausspunkt in physikalischen Koordinaten ausgedrückt werden:

$$\underline{\boldsymbol{x}}^{h}(\boldsymbol{\xi}_l) = \underline{\boldsymbol{N}}^{\mathrm{e}}(\boldsymbol{\xi}_l)\underline{\boldsymbol{x}}^{\mathrm{e}} \tag{10.2}$$

Die Verzerrung, welche diesem Punkt zugeordnet ist lässt sich (in Voigt-Notation) wie folgt bestimmen:

$$\underline{\boldsymbol{\varepsilon}}^{\mathrm{V}}(\boldsymbol{\xi}_l) = \underline{\boldsymbol{B}}^{\mathrm{e}}(\boldsymbol{\xi}_l)\underline{\boldsymbol{d}}^{\mathrm{e}} \tag{10.3}$$

Mithilfe des Elastizitätstensors $\underline{\mathbb{C}}^{V}$ lässt sich ebenso die Spannung bestimmen:

$$\underline{\boldsymbol{\sigma}}^{\mathrm{V}}(\boldsymbol{\xi}_l) = \underline{\mathbb{C}}^{V}\underline{\boldsymbol{\varepsilon}}^{\mathrm{V}}(\boldsymbol{\xi}_l) \tag{10.4}$$

Aufgabe 10.1: Gausspunkte

Die Subroutine `gauss04(l,lint,eg,wg)` gibt die Koordinaten `eg` und Wichtungsfaktoren `wg` bei Input des aktuellen Gausspunktes `l`. `lint` stellt dabei die Anzahl an Elementgausspunkten dar (lineares Dreieck t1: `lint`=1, quadratisches Dreieck t2: `lint`=3 oder lineares Viereck q1: `lint`=4).

a) Vervollständigen Sie `gauss04` um die Informationen aus Tabelle 10.3.

Aufgabe 10.2: Postprocessing

Die Subroutine `str04` berechnet die Verzerrungen `eps` und Spannungen `sig` aus den Elementknotenverschiebungen `du`. Dazu werden die Berechnungsschritte (10.2)-(10.4) durchgeführt. Die zugehörigen Programmvariablennamen sind Tabelle 10.4 zu entnehmen.

$\underline{\boldsymbol{x}}^{h}(\boldsymbol{\xi}_l)$	$\underline{\boldsymbol{N}}^{\mathrm{e}}(\boldsymbol{\xi}_l)$	$\underline{\boldsymbol{x}}^{\mathrm{e}}$	$\underline{\boldsymbol{d}}^{\mathrm{e}}$	$\underline{\boldsymbol{B}}^{\mathrm{e}}(\boldsymbol{\xi}_l)$	$\underline{\boldsymbol{\varepsilon}}^{\mathrm{V}}(\boldsymbol{\xi}_l)$	$\underline{\mathbb{C}}^{V}$	$\underline{\boldsymbol{\sigma}}^{\mathrm{V}}(\boldsymbol{\xi}_l)$
`xg`	`shp(3,*)`	`xl`	`du`	`bmat`	`eps`	`aa`*	`sig`

Tabelle 10.4: Zuordnung der Programm-Variablennamen zu den Berechnungsgrößen.

Hinweis: `aa`* ist die $[3 \times 3]$-Elastizitätsmatrix, deren Einträge zur x-y-Ebene gehören. Da die implementierten Elemente einen ebenen Verzerrungszustand beschreiben ist zu berücksichtigen, dass $\sigma_{33} \neq 0$ zusätzlich zu berechnen ist.

a) Ergänzen Sie `str04` um die Berechnung (10.2) der phys. Gausspunktkoordinaten

b) Ergänzen Sie `str04` um die Berechnung (10.3) der Verzerrungen

c) Ergänzen Sie `str04` um die Berechnung (10.4) der Spannungen

Aufgabe 10.3: Konvergenzstudie

a) Ersetzen Sie die vervollständigte Datei `elmt04.f` in dem Dateipfad `$FEAPPVHOME4_1/user` und updaten Sie das FEAPpv-Hauptprogram mit dem Befehl `make` (vgl Aufgabe 9.2).

b) Testen Sie `elmt04.f` mit einer Beispielrechnung (`I_cm`) und Vergleichen Sie die Ausgabe der Spannungen und Verzerrungen in der Outputdatei (`O_cm`) und die Contourplots mit denen der programmeigenen Elemente.

c) Ergänzen Sie Netzkonvergenzstudie aus Aufgabe 9.2c) mit den Ergebnissen des Viereckselementes. Plotten Sie die Verschiebung des Eckpunktes `A` über der Anzahl der Freiheitsgrade. Macht es einen qualitativen unterschied ob der Konvergenzplot über der Anzahl der Freiheitsgrade oder der Anzahl der Elemente erstellt wird?

Übung 11: Gauss-Quadratur

Die Formel für die numerische Integration mittels Gauss-Quadratur lautet

$$\int_{-1}^{1} f(\xi)\ \mathrm{d}\xi \approx \sum_{i=1}^{n} w_i\ f(\xi_i)\ . \tag{11.1}$$

Dabei sind w_i die Wichungsfaktoren und ξ_i die Gausspunkte.

Aufgabe 11.1: Quadratisches Viereckselement

Für das quadratische isoparametrische Viereckselement wird die Gaussintegrationsordnung $n = 3$ gewählt.

a) Wie viele Gausspunkte hat das Element dann?

b) Ist die Gaussintegration der Ordnung $n = 3$ für eine quadratische Funktion $f(\xi)$ (Polynomgrad $p = 2$) exakt? Welche Bedingung muss erfüllt sein?

Hinweis: Da bei Viereckselementen die Übertragung von einer auf zwei Koordinatenrichtungen direkt möglich ist reicht es in dieser Aufgabe eine Funktion $f(\xi)$ mit nur einer Koordinate zu betrachten.

c) Gibt es Gründe eine höhere Integrationsordnung als nötig zu verwenden? Erläutern Sie ihre Antwort anhand eines Beispiels.

Findet die Gaussintegration über das Intervall $[-1, 1]$ statt so lassen sich die Gausspunke ξ_i $(i = 1, \ldots, n)$ als Nullstellen des n-ten Legendrepolynoms bestimmen:

$$P_n(\xi) = \frac{1}{2^n\, n!}\ \frac{\mathrm{d}^n}{\mathrm{d}\xi^n}(\xi^2 - 1)^n \tag{11.2}$$

d) Wie lautet das Legendrepolynom 3. Ordnung?

e) Bestimmen Sie die entsprechenden Gausspunktkoordinaten ξ_i

Die Zugehörigen Wichtungsfaktoren w_i lassen sich über folgende Beziehung bestimmen:

$$w_i = \int_{-1}^{1} l_i(\xi)\ \mathrm{d}\xi \tag{11.3}$$

Dabei ist l_i das i-te Lagrangepolynom (Polynomgrad: $n - 1$) mit den Gausspunkten ξ_i als Stützstellen.

f) Bestimmen Sie die zu den Gausspunktkoordinaten gehörigen Wichtungsfaktoren w_i

Klausurvorbereitung

Aufgabe 1: Schwache Form und Idee der FEM

Gegeben sei das elastische Potential

$$\Pi = \frac{1}{2}\int_{\mathcal{B}} \boldsymbol{\varepsilon} \cdot \boldsymbol{\sigma} \, \mathrm{dv} - \int_{\mathcal{B}} \boldsymbol{u} \cdot \bar{\boldsymbol{b}} \, \mathrm{dv} - \int_{\partial \mathcal{B}_t} \boldsymbol{u} \cdot \bar{\boldsymbol{t}} \, \mathrm{da}$$

a) Wie lautet die zugehörige Variationsgleichung (schwache Form)?

b) Leiten Sie aus der schwachen Form die Euler Lagrange-Gleichungen (starke Form) für das Randwertproblem der Elastostatik her.

c) Wie gelangt man von der schwachen Form zur Elementsteifigkeitsmatrix $\underline{\boldsymbol{k}}_e$ und Elementlastmatrix $\underline{\boldsymbol{p}}_e$?

d) Wie lassen sich die Integrale in der diskretisierten schwachen Form berechnen?

e) Wie gelangt man zur Lösung des globalen Problems?

Aufgabe 2: Assemblierung

Gegeben sei das in Abbildung 1 dargestellte Mini-Randwertproblem bestehend aus linearen Dreieckselementen.

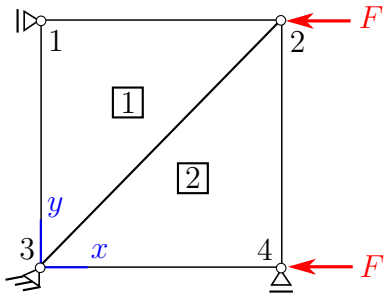


Abbildung 1

a) Erstellen Sie eine Skizze, in der alle globalen Knotenfreiheitsgrade (ohne Berücksichtigung der Randbedingungen) eingetragen sind.

b) Wie lauten die Einträge des globalen Lösungsvektors unter Berücksichtigung der Randbedingungen? Wie lautet der globale Lastvektor?

c) Wie lauten die Einträge der Elementsteifigkeitsmatrix $\underline{\boldsymbol{k}}^1$ und $\underline{\boldsymbol{k}}^2$, welche nach Berücksichtigung der Randbedingungen in die globale Steifigkeitsmatrix eingehen? Bestimmen Sie die Einträge der globalen Steifigkeitsmatrix $\underline{\boldsymbol{K}}$.

Fragenkatalog

Kontinuumsmechanik

- Was ist die Voigt-Notation? Welche Annahmen ermöglichen es diese zu verwenden?
- Wie lautet der Elastizitätstensor in Voigt-Notation?
- Was ist der Unterschied zwischen ebenem Verzerrungs- und ebenem Spannungszustand?

Grundkonzept der FEM

- Was sind die grundlegenden Schritte der Finite-Elemente Analyse?
- Leiten sie Elementsteifigkeitsmatrix und -lastmatrix her
- Wie werden Lagerungen in dem globalen Gleichungssystem berücksichtigt, wie werden die ensprechenden Randbedingungen noch genannt?
- Welche zwei verschiedenen Arten von Randbedingungen kennen Sie? Worin unterscheiden sie sich?
- Was passiert wenn Randbedingungen weggelassen werden?
- Welche verschiedenen Elementtypen kennen Sie?

Isoparametrisches Konzept

- Wie lauten die Ansatzfunktionen für das bilineare/quadratische Viereckselement? (Aufstellen der zugehörigen Lagrange-Polynome)
- Was besagt das Isoparametrische Konzept? Warum wird es verwendet?
- Wozu wird die Jacobideterminante benötigt? Wie wird sie bestimmt?

Numerische Integration

- Warum wird die Gauss-Integration benötigt?

Gegeben sei ein Polynom $f(\xi)$ mit dem Polynomgrad $n = 1/2/3/....$

- Wieviele Gausspunkte sind erforderlich damit die Integration exakt ist?
- Wie lauten die zugehörigen Gausspunkte und Wichtungsfaktoren? (entsprechende Formeln auswerten)
- Was passiert wenn eine höhere Integrationsordung als erforderlich verwendet wird?
- Wie lauten die Legendre-Polynome der Ordnung $n = 1/2/3/...$?

Konvergenz und Diskretisierungsfehler

- Wie lässt sich das Konvgenzverhalten verschiedener Elemente vergleichen? Skizzieren Sie einen beispielhaften entsprechende Konvergenzplot.
- Welche zwei verschiedenen Arten von A-Posteori Fehlerschätzern haben Sie kennengelernt, worin unterscheiden sich diese?
- Womit lässt sich die Qualität eines Fehlerschätzers ermitteln?
- Wird der Diskretisierungsfehler bei Erhöhung der Elementanzahl für das gleiche Randwertproblem größer oder kleiner? Begründen Sie Ihre Antwort.
- Was sind superkonvergente Punkte?
- Handelt es sich bei den Verschiebungen $\boldsymbol{u}^h$, wenn sie mit Lagrange-Ansatzfunktionen über die Knoten interpoliert um Funktionen, welche zwischen den Elementen kontinuierlich sind? Wie verhält es sich mit den zugehörigen Verzerrungen $\boldsymbol{\varepsilon}^h$?
- An welchen Punkten werden Verzerrungen und Spannungen im Postprocessing üblicherweise ausgewertet und warum?

Locking und gemischte Elemente

- Wann spricht man von inkompressiblem Material? Welche Probleme können auftreten?
- Was sind die Vorteile gemischter Elemente gegenüber standard Verschiebungselementen?
- Wie lautet das Hellinger-Reissner Variationsprinzip?
- Wie lautet das Hu-Washizu Variationsprinzip für das Q1P0 element?