

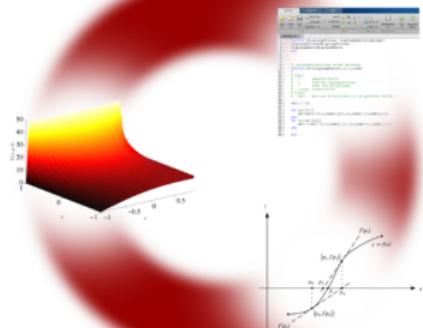
1. Einführung + Matlab

A Practical Course in Numerical Methods for Engineers

Barbara Wirthl, M.Sc.

Technische Universität München

Lehrstuhl für Numerische Mechanik



Ausbildung

- ▶ Seit Dez 2018: wissenschaftliche Mitarbeiterin am LNM
- ▶ 2012 – 2018: B.Sc., M.Sc. Maschinenwesen, TUM

Kontakt

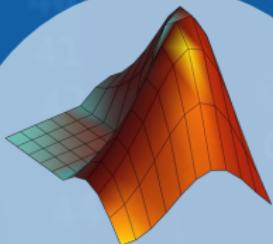
- ▶ E-Mail: barbara.wirthl@tum.de
- ▶ Im Forum *Fragen* im Moodle-Kurs

Studentische Arbeiten am LNM:

www.epc.ed.tum.de/lnm/student-projects

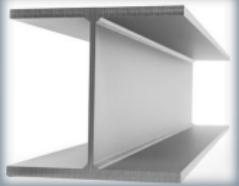
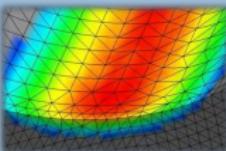
A Practical Course in Numerical Methods for Engineers

- Besuch parallel zur Vorlesung *Numerische Methoden für Ingenieure* möglich
- Flexible Zeiteinteilung
- 4 ECTS für Ergänzungsfächer



MATLAB
programmieren

Finite
Elemente



Wärmeleitungs-
gleichung



Teilnahme unter pingo.coactum.de → 396988

Pingo



Wer sind Sie?



In welchem Studiengang studieren Sie derzeit?

- ▶ Bachelor Maschinenwesen.
- ▶ Bachelor Ingenieurwissenschaften.
- ▶ Master Maschinenwesen bzw. anderer Master MW.
- ▶ Ich studiere an einer anderen School/Fakultät.

Überblick über den Inhalt des Moduls

Teil 1

1. Einführung + Matlab
2. & 3. Interpolation in 1D und 2D (2 Blätter)
4. Numerische Differentiation
5. Numerische Integration
6. Numerische Lösung von Anfangswertproblemen

Teil 2

7. & 8. Methode der Finiten Elemente:
stationäre und instationäre Wärmeleitung (2 Blätter)
9. Lösung von linearen Gleichungssystemen:
direkte und iterative Lösungsverfahren
10. Lösung von nichtlinearen Gleichungssystemen

Numerische Methoden



Haben Sie eine Vorlesung zu Numerischen Methoden inklusive eines Teils zu Finiten Elementen besucht?

- ▶ Ja, ich besuche parallel die Vorlesung *Numerische Methoden für Ingenieure* des LNM.
- ▶ Ja, ich habe die Vorlesung *Numerische Methoden für Ingenieure* des LNM bereits besucht.
- ▶ Ja, eine andere ähnliche Vorlesung.
- ▶ Nein.

Ablauf

Präsentation der Aufgaben:
Donnerstags 17:00 – 17:45 Uhr
MW2050 und Zoom (siehe Link im Moodle-Kurs)



Selbstständige Bearbeitung der Aufgaben in der folgenden Woche



Tutorsprechstunde:
Montags 10:00 Uhr – 12:15 Uhr im Red-Pool MW1264
Mittwochs 15:30 Uhr – 17:45 Uhr im Red-Pool MW1264



2 x im Semester: Überprüfung
voraussichtlich Mittwoch 7.12.2022 und Mittwoch 1.2.2023

MATLAB



Haben Sie MATLAB auf Ihrem privaten Laptop installiert?

- ▶ Ja.
- ▶ Nein und ich weiß nicht wie.
- ▶ Nein, aber ich weiß wie.



MATLAB

Haben Sie bereits Erfahrung mit Matlab?

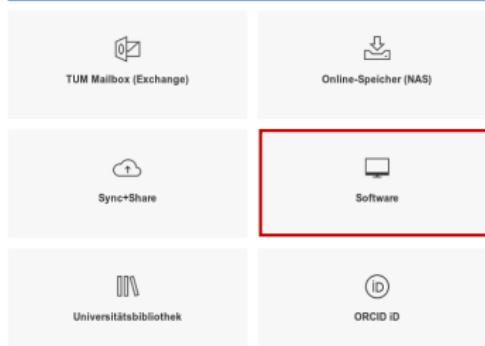
- ▶ Ja, ich habe bereits viele (auch umfangreichere) Dinge in Matlab implementiert.
- ▶ Ja, ein bisschen Erfahrung.
- ▶ Nein, aber dafür mit anderen Programmiersprachen.
- ▶ Nein, keine Erfahrung.

MATLAB

Die Bearbeitung der Aufgabenstellungen erfolgt mithilfe von Matlab.

- ▶ Verwendung von Matlab am eigenen Laptop
- ▶ Matlab Lizenz: Lizenzverteilung MathWorks-Rahmenvertrag
- ▶ Hilfe: <https://wiki.rbg.tum.de/Informatik/Helpdesk/MatlabInstallieren>

Andere IT Dienste



Soft- und Hardwareangebote der TUM / Soft- und Hardwareangebote der TUM

Hardware Informationen zum Bezug von Hardware verschiedener Hersteller finden Sie auf den Webseiten der Rechnerbetriebsgruppe in Garching.
Software Alle zentralen Software-Angebote mit Informationen zu Voraussetzungen und Zugang haben wir in einem Wiki zusammengestellt.
Weitere Produkte Weitere Produkte wie Matlab, ChemDraw, Adobe oder Origin können Sie beziehen, indem Sie sich hier eine spezielle E-Mail Adresse generieren, mit der Sie sich beim Anbieter registrieren können

Aufgabenblätter

Sie erhalten jede Woche ein neues Aufgabenblatt, das Sie selbstständig bearbeiten:

- ▶ Enthält bereits Lösungen (Zahlenwerte oder Diagramme), um den erstellten Code zu überprüfen
 - ▶ Lösungscodes werden nicht zur Verfügung gestellt
 - ▶ Vorgegebene durchnummerierte Funktionen, welche genau so erstellt werden sollen, um diese in späteren Aufgabenblättern wiederzuverwenden
- Je eine Matlab-Funktion in eigenem *.m-file



Diese Funktionen sollen automatisch mit dem Modultest auf das vorgegebene Ergebnis geprüft werden (weitere Hinweise siehe 1. Aufgabenblatt).

Matlab-Funktionen

Folgende Funktionen sollen bei der Bearbeitung dieses Aufgabenblattes erstellt werden, da diese für spätere Aufgabenblätter wiederverwendet werden sollen. Erstellen Sie die Funktionen in Matlab und speichern Sie diese in eigenen *.m-files ab.

Hilfe zu den Aufgabenblättern

- ▶ Matlab-Hilfe: <https://de.mathworks.com/help/matlab/>
- ▶ Matlab-Tutorials: <https://www.fsmb.de/fsmb/uebers-studium/it/matlab/tutorials/>
- ▶ Befehl 'help <functionname>' in der Matlab-Konsole
- ▶ Online Suchmaschinen
- ▶ Tutorsprechstunden: vorbereitet in die Tutorsprechstunden kommen
 - ✓ Fragen zum jeweils aktuellen Arbeitsblatt
 - ✓ Arbeitsblatt gelesen
 - ✓ Aufgaben in Matlab bearbeitet
 - ✓ falsches Ergebnis
 - ✓ selbstständige Fehlersuche (Eingrenzen des Fehlerbereiches)
 - ✓ alle verwendeten Funktionen mit Modultest getestet

Überprüfungen

2 Überprüfungen im Semester

1. Überprüfung: Arbeitsblätter 1 – 6
2. Überprüfung: Arbeitsblätter 7 – 10

- Matlab-Code zum Lösen der Aufgabenstellungen an den Computern im Red Pool
- Zum Lösen der Aufgabenstellungen sollten alle Arbeitsblätter selbstständig bearbeitet werden

Ablauf

- ▶ Bearbeitung an Rechnern des Red Pools
- ▶ Matlab-Funktionen werden je nach Aufgabenstellung ohne Beschreibung zur Verfügung gestellt
- ▶ Alle Unterlagen erlaubt, außer:
 - Keine Matlab-Codes in jeglicher Form (Matlab-Hilfe erlaubt)
 - Keine Pseudo-Matlab-Codes wie ‚Wenn x ist y, dann ...‘
- ▶ Keine externen Datenträger, keine E-Mails
- ▶ Internet zum Googeln erlaubt
- ▶ Matlab-Hilfe steht zur Verfügung

Aufgabenblatt 1

Aufgabe 1: Vektoren, Matrizen

Erstellen Sie ein Matlab-Skript, das folgende Operationen durchführt für

$$\mathbf{A} = \begin{pmatrix} 3 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 3 \end{pmatrix} \quad \mathbf{v} = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \end{pmatrix}$$

- $\mathbf{a} = \mathbf{v} \cdot \mathbf{v}$ Lsg.: 30.0
- $\mathbf{B} = \mathbf{v}\mathbf{v}^T$ Lsg.: [1, 2, 3, 4; 2, 4, 6, 8; 3, 6, 9, 12; 4, 8, 12, 16]
- $\mathbf{C} = \mathbf{AB}$ Lsg.: [3, 6, 9, 12; 6, 12, 18, 24; 9, 18, 27, 36; 12, 24, 36, 48]
- Berechnen Sie die Eigenwerte λ der Matrix \mathbf{C} mit der entsprechenden Matlab-Funktion.
Entspricht diese Lösung der exakten analytischen Lösung? Stellen Sie dazu qualitative Überlegungen an.

Lsg.: [-0.0000000000000002; 0.0; 0.0000000000000007; 90.0])

- ...

Aufgabenblatt 1

Aufgabe 2: Operatoren und Flusskontrolle, Funktionen

Erstellen Sie ein Matlab-Skript, das folgende Operationen durchführt:

- Erstellen Sie einen Zufallsvektor \mathbf{a} der Dimension 1000 mit Einträgen zwischen 0 und 1.
- Wenn der erste Eintrag $a_1 \geq 0.5 \rightarrow$ Matlab-Ausgabe ' $a1 \geq 0.5$ ', sonst ' $a1 < 0.5$ '
(Hinweis: if).
- Ermitteln Sie die Anzahl n_{05} der Einträge $a_i \geq 0.5$ im Vektor \mathbf{a} (Hinweis: for,if).
- Ermitteln Sie den ersten Eintrag (Index i & Wert a_i) im Vektor \mathbf{a} , für den gilt $0.499 \leq a_i \leq 0.501$. Falls kein Element existiert, welches das Kriterium erfüllt \rightarrow Matlab-Ausgabe 'Kein Element $0.499 \leq a_i \leq 0.501$ ' (Hinweis: while,if).
- Berechnen Sie den Wert von $n!$ für ganzzahlige n . Erstellen Sie dazu eine eigene Funktion **Fkt. A** (siehe unten). Die Funktion soll in einem eigenen *.m-file abgespeichert werden.
Testen Sie die Funktion für $n = 0$ und $n = 5$.

Bearbeitung des Aufgabenblattes

Erstellen eines *.m-files: Blatt1.m bzw. eines eigenen *.m-files für jede Aufgabe.

```
% Skript, das alle Aufgaben des ersten Arbeitsblattes berechnet!
clc;
clear all;
close all;

addpath(' ../../m-files/');

format long;

Aufgabe1(); % Berechnungen fuer Aufgabe 1
Aufgabe2(); % Berechnungen fuer Aufgabe 2
Aufgabe3(); % Berechnungen fuer Aufgabe 3
mtest(); % Modultest fuer Aufgabe 4 durchfuehren

% Berechnet alle Aufgabenstellungen aus Aufgabe 1
function Aufgabe1()
    %
    %
end
```

- ▶ Erstellen Sie für jedes Aufgabenblatt einen Ordner, der alle *.m-files für dieses Arbeitsblatt enthält, z.B. '/PNuME/Blatt1/m-files/Blatt1.m '
- ▶ Erstellen Sie für das gesamte Semester einen Ordner, der alle *.m-files der **Matlab-Funktionen** enthält, z.B. '/PNuME/m-files/fakultaet.m '

Debugger

Breakpoints setzen:

```
5      function Aufgabe1()
6
7 -    a = 1;
8 -    b = 2;
9 -    c = a + b;
10 -   d = a * b;
11
12 - end
```

Im Workspace überprüfen, ob Sie das berechnen, was Sie denken:

Workspace - Aufgabe1	
Name	Value
a	1
b	2
c	3

Aufgabenblatt 1

Aufgabe 3: Plots

Werten Sie eine Funktion an den gewünschten Punkten aus und plotten Sie diese (kein direktes Plotten analytischer Funktionen).

- 2D–Plot: ...
- 3D–Quadplot: Erstellen Sie ein Programm, welches beliebige 3D–Flächen im Raum plottet, wobei die Flächen aus einer beliebigen Anzahl an Vierecken bestehen können. Als erster Schritt soll die allgemeine Funktion **Fkt. 0** zum Plotten beliebiger Vierecke erstellt werden. Diese soll alle Vierecke in jeweils zwei Dreiecke unterteilen und dann diese mithilfe der Matlab-Plot-Funktion **trisurf** plotten.

Als zweiter Schritt soll die Funktion verwendet werden, um folgende 3D–Fläche zu plotten: Gegeben ist ein Gitter bestehend aus 4 Vierecken mit den Eckkoordinaten $x = -1, 0, 1$ und $y = -1, 0, 1$. Die z -Koordinate in den Ecken der Vierecke ist durch die Funktion $f(x, y) = x^2 + y^2$ gegeben.

Quadplot: Beispiel zu Aufgabe 3

```
function quadplot(nodes,elements,sol)
```

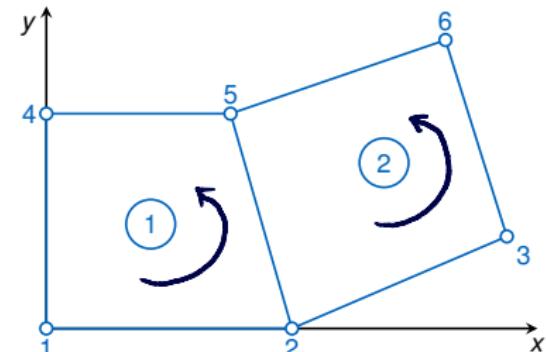
Knoten: nodes

```
nodes = zeros(6,2);
nodes(1,:) = [0,0];
⋮
nodes(6,:) = [x6,y6];
```

Elemente: elements

Anzahl Elemente) Knoten pro Element

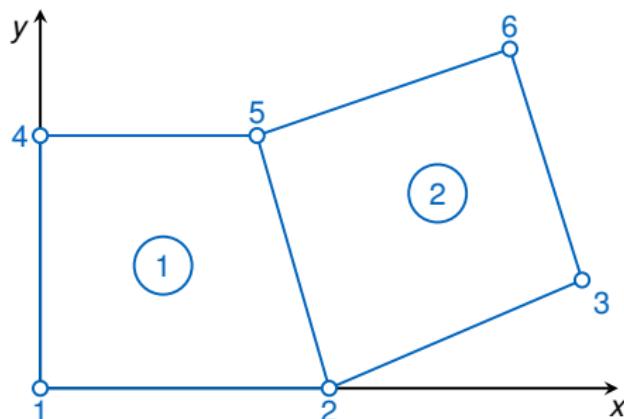
```
elements = zeros(2,4);
elements(1,:) = [1,2,5,4];
elements(2,:) = [2,3,6,5];
```



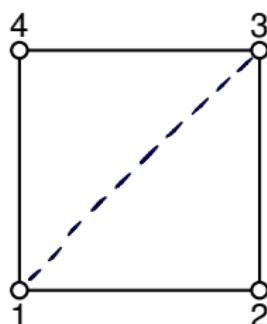
z-Werte: sol

```
sol = zeros(6,1);
sol(1,1) = x12 + y12;
⋮
```

Quadplot: Beispiel zu Aufgabe 3



Standardelement:



Knoten pro Element in 2 Dreiecke unterteilen:

Dreieckskonnektivität

$$T = [1, 2, 3; 3, 4, 1]$$

→ trisurf(T, x, y, z)

Aufgabenblatt 1

Aufgabe 4: Modultests

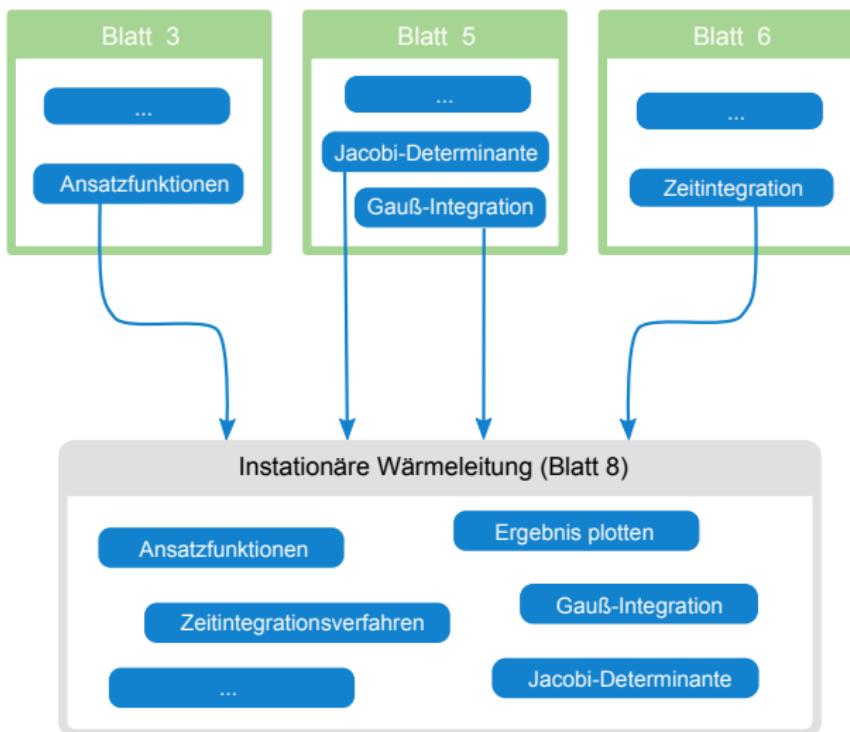
Erstellen Sie ein Matlab-Skript (und speichern Sie es in einem eigenen *.m-file ab), das den Rückgabewert von Funktionen mit einem vorgegebenen Ergebnis auf einen maximalen absoluten festgelegten Fehler (Toleranz) prüft. Ihr Skript soll ausgeben, welche Funktionen den Test bestehen bzw. welche nicht. Tragen Sie als erste Funktion `facultaet(n)` in das Skript ein und testen Sie das Ergebnis für $n = 0$ und $n = 5$ mit einer Toleranz von 10^{-12} .

Hinweis:

Dieses Skript soll im Laufe des Semesters erweitert werden, sodass alle Matlab-Funktionen, die während des Semesters erstellt werden, geprüft werden können. Ergebnisse können Skalare, Vektoren oder Matrizen sein. Die genaue Gestaltung des Skripts ist Ihnen selbst überlassen.

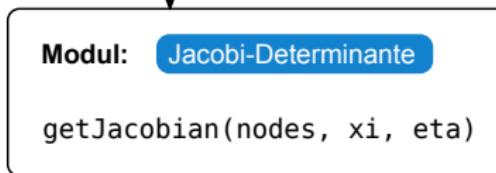
Modultest

Jede Matlab-Funktion einzeln testen:



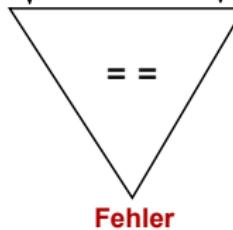
Modultest: Beispiel

Input: nodes = [2,1;4,1;4,3;2,2]
 $x_i = 0.577$
 $\eta = 0.577$



Erwarteter Output:
 $detJ = 0.89425$

Berechneter Output:
 $detJ = 0.8$



Modultest

Command Window

```
#####
##### Modultest #####
001: Testing function <Fkt A: facultae_0> ... passed (= 1, tol = 1e-12)!
002: Testing function <Fkt A: facultae_5> ... passed (= 120, tol = 1e-12)!
```

003: !!! Testing function <Fkt A: facultae_0,1,2,3> ... failed ([1 1;2 6] ~= [1.01 1.02;2.0001 6], diff=[-0.01 -0.02;-0.000100000000)

```
004: Testing function <Fkt I: linquadref(0,0)> ... passed (= [0.25;0.25;0.25], tol = 1e-12)!
```

005: Testing function <Fkt I: linquadref(0.577,-0.577)> ... passed (= [0.16676775;0.62173225;0.16676775;0.04473225], tol = 1e-12)!

006: Testing function <Fkt II: linquadriderivref(0,0)> ... passed (= [-0.25 -0.25;0.25 -0.25;0.25 0.25;-0.25 0.25], tol = 1e-12)!

007: Testing function <Fkt II: linquadriderivref(0.577,-0.577)> ... passed (= [-0.39425 -0.10575;0.39425 -0.39425;0.10575 0.39425;-0.39425,-0.39425])

008: Testing function <Fkt III: gx(3)> ... passed (= [-0.774596669241483 0.774596669241483], tol = 1e-12)!

009: Testing function <Fkt IV: gw(3)> ... passed (= [0.5555555555555556 0.8888888888888889 0.555555555555556], tol = 1e-12)!

010: Testing function <Fkt V: gx2dref(1)> ... passed (= [0 0], tol = 1e-12)!

011: Testing function <Fkt V: gx2dref(2)> ... passed (= [-0.577350269189626 -0.577350269189626;-0.577350269189626 0.577350269189626)

012: Testing function <Fkt VI: gx2dref(1)> ... passed (= 4, tol = 1e-12)!

013: Testing function <Fkt VI: gx2dref(2)> ... passed (= [1;1;1], tol = 1e-12)!

014: Testing function <Fkt VII: getxPos([2,1;4,1;4,3;2,2],0.577,-0.577)> ... passed (= [3.577;1.37826775], tol = 1e-12)!

015: Testing function <Fkt VIII: getJacobian([2,1;4,1;4,3;2,2],0.577,-0.577), J> ... passed (= [1 0;0.10575 0.89425], tol = 1e-12)!

016: Testing function <Fkt VIII: getJacobian([2,1;4,1;4,3;2,2],0.577,-0.577), detJ> ... passed (= 0.89425, tol = 1e-12)!

017: Testing function <Fkt VIII: getJacobian([2,1;4,1;4,3;2,2],0.577,-0.577), testinvJ> ... passed (= [1 0;-0.118255521386637 1.1182)

018: Testing function <Fkt IX: OST(0.5,0.2,1,1,[1,4,1,5],[1,7,1,8],[2,0]), LHS> ... passed (= 0.96, tol = 1e-12)!

019: Testing function <Fkt IX: OST(0.5,0.2,1,1,[1,4,1,5],[1,7,1,8],[2,0]), RHS> ... passed (= 2.85, tol = 1e-12)!

020: Testing function <Fkt X: AB2(0.2,1,1,[1,5,1,6],[1,8,1,9],[2,0,2,1]), LHS> ... passed (= 1.1, tol = 1e-12)!

021: Testing function <Fkt X: AB2(0.2,1,1,[1,5,1,6],[1,8,1,9],[2,0,2,1]), RHS> ... passed (= 3.114, tol = 1e-12)!

022: Testing function <Fkt XI: AM3(0.2,1,1,[1,4,1,5,1,6],[1,7,1,8,1,9],[2,0,2,1]), LHS> ... passed (= 0.9833333333333333, tol = 1e-12)!

023: Testing function <Fkt XI: AM3(0.2,1,1,[1,4,1,5,1,6],[1,7,1,8,1,9],[2,0,2,1]), RHS> ... passed (= 2.894, tol = 1e-12)!

024: Testing function <Fkt XII: BDF2(0.2,1,1,1,4,1,7,[2,0,2,1]), LHS ... passed (= 1.37, tol = 1e-12)!

025: Testing function <Fkt XII: BDF2(0.2,1,1,1,4,1,7,[2,0,2,1]), RHS ... passed (= 3.585, tol = 1e-12)!

026: Testing function <Fkt XIV: solveGauss([10,0,2,1;3,4,4,1;8,4],[1;1;2])> ... passed (= [0.051280512820513;0.275641025641026;-0.

027: Testing function <Fkt XV: solveG([10,0,2,0,10;0,2,0,40,0,8,0;10,0,8,0,60,0],[1;0;1;0;2,0],[0;0;0;0;0,0],1e-7,1000)> ... passed

028: Testing function <Fkt XVI: solveCG([10,0,2,0,10,0;2,0,40,0,8,0;10,0,8,0,60,0],[1;0;1;0;2,0],[0;0;0;0;0,0],1e-7,1000)> ... passed

029: Testing function <Fkt XVII: evaluate_stat([0;0;1;0;1;2;0,2],gx2dref(3),gx2dref(3)), elemat> ... passed (= [40 -28 -20 8;-28 40)

030: Testing function <Fkt XVII: evaluate_stat([0;0;1;0;1;2;0,2],gx2dref(3),gx2dref(3)), elevect> ... passed (= [0;0;0;0], tol = 1e-0)

031: Testing function <Fkt XVIII: assemble([1,2,3,4;5,6,7;8;9,10,11,12;13,14,15,16],[17;18;19;20],eye(5,5),ones(5,1),[5,3,1,2]), sys

032: Testing function <Fkt XVIII: assemble([1,2,3,4;5,6,7;8;9,10,11,12;13,14,15,16],[17;18;19;20],eye(5,5),ones(5,1),[5,3,1,2]), rhs

033: !!! Testing function <Fkt XIX: assignDBC([12,12,10,0,9;15,17,14,0,13;7,8,7,0,5;0,0,0,1,0;3,4,2,0,2],[20;21;19;1;18],[2,-7;5,-2]), s

034: !!! Testing function <Fkt XY: assignDRC([12,12,10,0,15;17,14,0,13;7,8,7,0,5;0,0,0,1,0;3,4,2,0,2],[20;21;19;1;18],[2,-7;5,-2]), r

Und los...

Erste Tutorsprechstunden:

Montag 24.10. 10:00 – 12:15 Uhr MW1264
Mittwoch 26.10. 15:30 – 17:45 Uhr MW1264

Nächstes Aufgabenblatt:

Donnerstag 27.10. 17:00 – 17:45 Uhr MW2050 + Zoom