LAK-CompMath: Woche 1

Leitfaden Intro

1. Begrüßung und eigene Vorstellung (Wie erreichbar?)
2. zum Übungsablauf:
   - Übungsaufgaben stehen als pdf auf der LV-Homepage.
   - Anwesenheitspflicht in allen Übungen (75% Quote).
   - In Abgabewochen erläutern Studenten ihre Lösungen zu den Aufgaben. Tafelbewertung.
   - Tutoriumstermin am Montag 17:00-18:30 online.
   - Aufgabenabgabe über Moodle.
3. Endnote, siehe Modus und Fristen auf der LV-Homepage
   - Sobald eine Übungsaufgabe abgegeben wurde, ist eine Note zu vergeben!
4. Zeigen am Computer:
   - Einloggen
   - Starten von Matlab
   - ---> Studenten starten ebenfalls Matlab
5. Kurze Einführung der Matlab-IDE:
   - Begriffe: Command Window, Editor, Workspace (zeigen!)
   - Hilfe in Matlab:
     - help; help matlab/graph2d; help sin
     - doc; doc matlab/graph2d; help sin
6. Matlab-Kommandos in Command Window eingeben (Lehrer+Studs): analog zu Zeilen 3-23 aus ue_0_demo.m
   - Ausgabeunterdrückung mit „ ; “
   - Änderung der Variablenwerte im Workspace beobachten (whos)
   - Direkte Änderung von Variablenwerten in Workspace (draufklicken und Wert editieren)
   - Mechanismus: save → clear → load und dessen Auswirkung auf Workspace zeigen (nur kurz)

1. Symbolische Methoden in Matlab

V_21/v_1_a.m

1.1. Deklaration symbolischer Variablen/Funktionen:

- syms a b c; % deklariere symbolische Variablen
- f(a,b,c) = a+b+c % symbolische **Funktion**

1.2. Ersetzen von Variablen durch Zahlenwerte oder Ausdrücke

- f(3,b,c) % Ersetze in f die Variable a durch 3
- f(b,b,c) % Ersetze in f die Variable a durch (bereits als symb. def.) Variable b
- f('x',b,c) % Ersetze in f die Variable a durch (noch nicht als symb.) Variable x
- f((b+c)^2,b,c) % Ersetze in f die Variable a durch den Term (b+c)^2
- f(1,2,3) % Ersetze in f die Variable die Variablen durch die Werte/Terme
- Umwandeln symbolischer Zahlenwerte in reelle Zahlen: eval(...)

1.3. Manipulation algebraischer Ausdrücke

- syms a b; % deklariere symb. Variablen
- f = a^2-2*a*b+b^2 % symb. Funktion
- fe = factor(f) % faktorisieren → Vekor der Faktoren
- prod(fe) % ausmultiplizieren der Faktoren
- fs = simplify(f) % (bestmoeglich) vereinfachen, zeigt alle Versuche
- expand(fs) % ausmultiplizieren des Ausdruckes

1.4. Löse einzelne algebraische Gleichungen

Löse x^2 = 16 <==> x^2-16 **= 0**

- symbolisch (*alle* Lösungen):
  - syms x;
  - y = x^2-16;
  - solve(y, x) % Loest y(x) = 0
- symbolisch mit Variableneinschränkung
  - assume(x>0)
  - a0 = solve(f==0,x)
  - assumeAlso(x<10)
- numerisch lösen (*eine* Lösung) mit symbolischer Funktion → symbolische Zahl:
  - vpasolve(y)
- numerisch lösen (*eine* Lösung) mit numerischer Funktion → numerische Zahl:
  - y_hand = matlabFunction(y); % symb. → numer. Funktion
  - % y_hand =@(x) x^2-16 % direkte definition der num. Funktion
  - fzero(y_hand, -1),
  - fsolve(y_hand, 1) [Startwert für Iteration !] % nur im Optimierungspaket enthalten.
- algebraisch==> numerisch
  - f_hand = @(x)subs(y,x); % Functionhandle aus symbolischer Fkt. erzeugen
  - f_hand = matlabFunction(y) % Functionhandle aus symb. Fkt. erzeugen
  - sol = fzero(f_hand, 1)

Umstellen von Gleichungen nach einer Variablen <==> Loese nach einer bestimmten Variablen symbolisch auf

Stelle y = 2*x^2+4 nach x um:

- syms x y;
- f = 2*x^2+4-y % Notwendig: f(x,y) = 0
- aa = solve(f,x)
- pretty(aa) % Schönere Ausgabe

1.5. Löse Gleichungssysteme (symbolisch)

Lineares Gleichungsystem: a + b = 3 und a + 2*b = 6

- symbolisch: <==> a + b - 3 = 0 und a + 2*b -6 = 0
  - syms a b;
  - f(1) = a + b - 3;
  - f(2) = a + 2*b - 6;
  - [A,B] = solve(f(1),f(2))
- symbolisch mit Parameter
  - syms a b c;
  - f(1) = a + b - c;
  - f(2) = a + 2*b - c^2;
  - [A,B] = solve(f(1),f(2), a,b)
- numerisch: x = K\f

Nichtlineares System:

3a^2 + b^2 = 1 und a + b = 1 <==> 3a^2 + b^2 – 1 = 0 und a + b - 1 = 0;

- symbolisch (*alle* Lösungen)
  - syms a b;
  - f(1) = 3*a^2 + b^2 - 1;
  - f(2) = a + b - 1;
  - [A,B] = solve(f(1),f(2))
  - pretty([A,B])
- symbolisch mit Parameter
- numerische Lösung (*eine* Lösung)
  - x_0 = [0.5, 0.6]; % Startwert für die Iteration
  - sol = fsolve(@(x)[3*x(1)^2 + x(2)^2 - 1; x(1) + x(2) - 1], x_0)

1.6. Integrieren, Differenzieren, Grenzwert:

Diff./Integrieren in 1D

- syms x;
- f = x^2 - 3*x + 4
- diff(f) % oder diff('x^2 - 3*x + 4')
- int(f) % unbestimmtes Integral
- int(f,x)
- int(f,0,1) % bestimmtes Integral
- int(f,x,0,1)

Int+Diff+Vereinfachen

- syms x
- f = exp(x)*cos(x)
- F = int(f,x)
- Fx = diff(F,x)

- Fx-f
- simplify(Fx-f)
- ezplot(F); hold on; ezplot(f)

Diff./Integrieren im höherdimensionalen:

```
syms a b
f = [a^2 + b^2 - 1, a + b - 1]
Jac = jacobian(f);
f1_a = diff(f(1),a)          % 1. Ableitung 1. Funktion bzgl. a
f1_a = diff(f(1),2)          % 2. Ableitung 1. Funktion bzgl. a
int(f(1))                    % unbestimmtes Integral bzgl. letzer Variable
int(f(1),a)                  % unbestimmtes Integral bzgl. Variable a
int(f(1),a,1,2)              % bestimmtes Integral bzgl. Variable a
```

Differentialoperatoren: gradient, laplacian, divergence, curl

Grenzwerte:

```
syms x a k;
pretty(limit((1 + a/x)^x, x, inf))              % Grenzwert
pretty(symsum(x^k/sym('k!'), k, 0, inf))    % Grenzwert von Reihen
pretty(symsum(1/k^2, k, 1, inf))            % Grenzwert von Reihen
```

1.7. Grafik mit symbolischen Funktionen

```
clear, clf
syms x real        %assume(x,'real')
g(x) = (x+x*cos(x))/(2*exp(x)-log(x));      % Symb. Funktion g(x)
interval = [0,10];
ezplot(g ,interval); %fplot(g ,interval)
g1 = diff(g,x);                    % 1. Ableitung
hold on
ezplot(g1 ,interval); %fplot(g1 ,interval)
g2 = diff(g,2,x);                  % 2. Ableitung
ezplot(g2 ,interval); %fplot(g2 ,interval)
legend("g(x)","g'(x)","g"(x)")       % Beschriftung der Plots
title("Demonstration 1D Funktion")
saveas(gcf,'g_1d.jpg')             % Speichere aktuelle Grafik in File
hold off
```

Siehe auch Visualisierung in 2D mit:

```
close all          % Schließen aller Grafikfenster
fsurf / ezsurf     % 2D Grafik in Matlab/Octave
```