CompMath-Vorlesung 4. Okt. 2024

Begrüßung, Vorstellung

Prof. Haase (Heinrichstr 36, IV. Stock)

- „Algorithmische Numerik“
- Studium: Lehrer Mathe/Physik + Mathematik
    - (Betreuer keine Zeit → Bibliothek → Diss gelesen → implementiert → Studien-/Jobangebot)
- www → Schedule, gundolf.haase@uni-graz.at,
- wissenschaftliches Rechnen: Mathematik+Computer+Anwendungen (VOEST, ENGEL, AVL, MAGNA, NUMECA, vif, RCPE, TUG)

Homepage der LV vorstellen, Moodle

- Mathematiker: Matlab+python/sage (3h)
- Sommersemester C++

Organisation der LV:

- Übungsaufgaben + Abgabe + Kreuzln dafür
    - Abgabe via Moodle
- Übung: Anwesenheitspflicht in Kontrollwoche
- Vorlesung: alle Gruppen zusammen

Benotung:

- Kreuzln für Übungsaufgaben + Tafel [mind. 50%]
- test matlab + test sage [mind. 50%]
- Gesamtnote: Übung und Test(s) Prozente gleichgewichtet

Fragen?

Inhalt der LV

Matlab [Prof. Haase, KFU]:

- MATrix LABoratory
- numerische Mathematik, Modellierung, Optimierung, symbolische Berechnungen, angewandte Mathematik
- Erlaubt mathematische Experimente, erlaubt schnelles Ausprobieren/Überprüfen/Visualisieren.
- closed source Software (Zugriff siehe LV-www), Octave 6.3.0 (Win)
- riesiger Funktionsumfang, viele, weitere Bibliotheken verfügbar
- wird sehr häufig auch von Firmen und Ingenieuren benutzt
- exzellente Dokumentation

python/sage [Prof. Lehner, TUG]:

- mathematische Software basierend auf Python (Tutor), Open Source

- Algebra, Zahlentheorie, Graphentheorie, symbolische Berechnungen (, Numerik)
- es gibt eine Dokumentation

Inhaltsverzeichnis:

- Variablen, (einige) Funktionen in Matlab, Matlab-Skripten
- arithmetische Operationen, mathematische Funktionen, Vektoren, einfache Grafik
- Matrizen, Vektoren und deren math. Operationen
- strukturierte Programmierung (Intro)
- einfache 3D-Körper und Flächen im Raum visualisieren
- symbolisches Rechnen mit Matlab
- Aufgabenlösen mit Matlab
- sage ….

Matlab- Was ist das?

Quick start

Start von Matlab

- *Command Window*, *Workspace*, *Editor*

Motivierende Beispiele (alles Matlab-Skripte):

- Abrollen_Film/film.m [Brunnsteiner: MAGNA → Uni Leoben → AVL]
- V_13/Inkugel_Tetraeder
- V_13/bsp_14 (Bildbearbeitung), V_13/bsp_34 (Brownsche Bewegung), V_13/bsp_44.m (Nullstellen einer Funktion), V_13/bsp_30 (Fläche zwischen 2 Kurven)
- ziege_wiese

Matlab-Skript: *.m – File im *Editor* zu bearbeiten und zu starten

Einige Helferlein zur Orientierung (meist im Command Window benutzt):

- help sin
- doc sin
- clear
- clc
- pwd
- ls
- who, whos

Matlab als Taschenrechner (→ Übung)

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 % symbolischer **Ausdruck**
- g(a,b,c) = a+b+c % symbolische **Funktion**

1.2. Ersetzen von Variablen durch Zahlenwerte oder Ausdrücke

- subs(f, a, 3) % Ersetze in f die Variable a durch 3
- subs(f, a, b) % Ersetze in f die Variable a durch (bereits als symb. def.) Variable b
- subs(f, a, 'x') % Ersetze in f die Variable a durch (noch nicht als symb.) Variable x
- subs(f, a, (b+c)^2) % Ersetze in f die Variable a durch den Term (b+c)^2
- subs(f, [a,b,c], [1,2,3]) % Ersetze in f die Variable die Variablen durch die Werte/Terme
- Bei symbolischen Funktionen: g(1,2,3)
- 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

Beispiel Weltrekordentwicklung 100m-Sprint (M/F)

- v_1a.m
- Daten aus www_1 und www_2
- Vektor
- save/load (*.mat !!!)
- plot
- title, legend, xlabel, ylabel
- saveas ← doc saveas

Funktionen in Matlab (Q & D)

- **Input** ---> **Funktion** → **Output**
- Bsp.: **a** = **sqrt**(**5.2**)
- Liste Inputparameter (, , ,)
- Liste Outputparameter [, , ,]
- Zeigen: Funktion zum Umrechnen von Celsius in Kelvin und Grad Fahrenheit (c2kf.m)
- k = c+2.73.15; f = 9/5*c + 32;
- Anwender der Funktion
  - im *Command Window*
  - in Skript mit Skalaren (celsius.m)
  - in Skript mit Vektoren (celsius_vek.m), evtl. Umstellung auf .* in Multiplikation
- Dokumentation in Funktion c2kf: help c2kf --> Dokumentiert eigene Funktionen!

Maschinen-Epsilon

- 1+eps> 1
- eps(‘single‘), eps(‘double‘)