

Programmierpraktikum im WS 2011/2012

Denny Otten, V5-142, dotten@math.uni-bielefeld.de

1. Erste Schritte mit Matlab

1 Inhalte des Programmierkurses

Grundlagen

- Einführung in die Syntax von Matlab
- Skripte, Prozeduren, Funktionen
- Kontrollstrukturen: if, while, for ...
- Datenstrukturen: Matrizen, Listen, Strukturen, ...
- Datenvisualisierung mit Matlab: plot, mesh, surf ...
- Eingabe/Ausgabe von Dateien
- Vektorisierung: effizienter Umgang mit Arrays

Fortgeschrittenere Techniken

- Fehlersuche mit dem Debugger
- Fehlerbehandlung: try
- Geschwindigkeitsoptimierung mit dem Profiler
- Verbindung mit der Symbolic Toolbox
- Erstellung von Animationen/Movies
- Erstellung graphischer Oberflächen mit GUIDE

Voraussetzungen für die Vergabe von Schein/Punkten

- aktive Mitarbeit während des Kurses
- erfolgreiche Bearbeitung der Pflichtaufgaben

2 Vorbereitungen

Einloggen

elektronisch eingetragen im eKVV

Account wurde angelegt: Anfangsbuchstabe Vorname + sieben Buchstaben des Nachnamens.
In diesem Fall (**und nur in diesem Fall!**) gib passwd in einer Linux-Kommandozeile (Shell)
ein und ändere Dein Passwort!

sonst

Fülle Accountantrag aus und sprich einen der Veranstalter an!

Das Passwort lautet in beiden Fällen: MATLABWS1112.

Die Shell

Bevor wir mit Matlab beginnen, wollen wir uns ganz kurz die Linux-Kommandozeile (Shell) anschauen. Wenn man sich an die Shell gewöhnt hat, ist sie mindestens genauso leicht zu bedienen wie ein typischer Dateibrowser, jedoch viel mächtiger. Du startest die Shell, indem Du im Startmenü (in der Leiste links unten) den Punkt **Konsole** auswählst (eventuell unter dem Punkt **System** versteckt). Probiere in dem sich öffnenden Fenster folgenden Befehle aus:

```
$ ls          listing – der Inhalt des aktuellen Verzeichnisses wird angezeigt
$ pwd        print working directory – Verzeichnis, indem Du Dich befindest
$ cd ..     change directory – .. ist das übergeordnete Verzeichnis
$ pwd       Was liefert pwd nun?
$ cd ..     jetzt solltest Du Dich im Wurzelverzeichnis befinden
$ ls        im Hauptverzeichnis sind einige Dateien
$ man ls    zeigt die Hilfe zu ls an, beende die Hilfe mit q
$ cd homes  somit wechselst Du wieder in das Verzeichnis homes
$ cd        ohne Argument schickt Dich cd direkt wieder in Dein Home-Verzeichnis
$ mkdir MatlabKurs  erzeuge ein neues Unterverzeichnis mit dem Namen "MatlabKurs"
$ cd MatlabKurs  und wir wechseln auch gleich hinein
```

Damit solltest Du jetzt – auch ohne Maus – in den Verzeichnissen navigieren können. Du kannst von der Konsole aus jedes Programm starten, Dateien sortieren, bearbeiten, löschen, drucken, anschauen,...

Um ein Programm zu starten gibst Du einfach den Namen des Programms in der Kommandozeile ein:

```
$ firefox &  startet den Webbrowser Firefox
```

das angehängte **&** bewirkt, dass die Eingabeaufforderung frei bleibt, Du also in andere Verzeichnisse wechseln, andere Programme starten,... kannst.

Unter Unix/Linux hat jedes laufende Programm (genauer, jeder laufende Prozess) eine Nummer, die PID (Prozess-ID). Während **firefox** läuft kannst Du Dir mit

```
$ pidof firefox-bin
```

die Prozess-ID von Firefox anzeigen lassen. Dieses ist zum Beispiel interessant, wenn der Prozess nicht mehr reagiert, dann kannst Du ihn nämlich mit **kill <PID>** "abschießen" (dabei ist PID natürlich die Prozessnummer). Du kannst Dir mit dem Befehl **top** die Prozesse anzeigen lassen, die momentan die meisten Ressourcen verbrauchen. Wenn Du **top** aufgerufen hast, erhältst Du durch die Eingabe von **?** eine Hilfe über die Tasten, mit denen Du **top** steuern kannst. Du beendest das Programm wieder mit Drücken der Taste **q**.

Die Shell kann noch sehr viel mehr, insbesondere kann man auch in ihr Programmieren (erinnert ein wenig an die **.bat**-Dateien aus alten DOS-Zeiten), aber für uns soll obiges als Bedienungshinweise genügen.

Eine (sehr kleine) Auswahl weiterer Befehle in der Shell ist unten aufgeführt. Ganz besonders seien Dir die Befehle **man** und **info** ans Herz gelegt, mit denen Du Hilfe bekommst.

Ferner findet man zum Beispiel unter http://linuxcommand.org/learning_the_shell.php und an vielen weiteren Stellen im Internet viele weitere Hinweise und Tipps zur Shell und auch zur Shell Programmierung.

Aufgabe 1. Erzeuge in Deinem Homes-Verzeichnis (**/homes/DeinName**) – falls Du es nicht schon oben gemacht hast – ein Verzeichnis **MatlabKurs**, wechsel hinein und erzeuge hier noch ein Unterverzeichnis **01**, wechsel wieder hinein. Hier wollen wir gleich Matlab starten.

Prozesse und sonstiges

man Kommando	(manual) zeigt eine Hilfe zum Kommando an
program &	startet das Programm program im Hintergrund, die Kommandozeile (Shell) bleibt dann frei zum weiterarbeiten.
ctrl-z	Stoppt ein im Vordergrund laufendes Programm
bg	(background) schickt das Programm in den Hintergrund
fg	(foreground) holt das Programm wieder in den Vordergrund
ps aux	zeigt alle aktuell laufenden Prozesse an
ps u	zeigt die laufenden Prozesse des Benutzers an
top	zeigt eine laufend aktualisierte Liste aktueller Prozesse an Mit ? (help) wird eine Hilfe zu top angezeigt, mit k (kill) lassen sich laufende Prozesse beenden
ps grep progname	zeigt die Daten (insbesondere die PID) des Prozesses an, der zum Programm progname gehört (z.B. ps grep matlab)
pidof progname	zeigt die PID von program an (Tab vervollständigt progname)
kill PID	beendet den Prozess mit der Prozess Id PID die PID wird von ps oder top angezeigt
apropos Keyword	liefert Kurzbeschreibungen von Kommandos, die im Zusammenhang mit dem Stichwort "Keyword" stehen
passwd	Ändert das eigene Passwort
emacs datei	Öffnet die Datei datei mit dem Editor emacs

Dateioperationen

ls	(listing) zeigt die Dateien im aktuellen Verzeichnis an
ls -l	wie ls mit ausführlichen Informationen
cd verzeichnis	(change directory) wechselt in das Verzeichnis verzeichnis
cp quelle ziel	(copy) kopiert die Datei quelle nach ziel
cp -r quelle ziel	(copy recursively) kopiert das Verzeichnis quelle nach ziel
mv quelle ziel	(move) verschiebt die Datei oder das Verzeichnis quelle nach ziel
rm datei	(remove) löscht die Datei datei
mkdir verzeichnis	(make dir) erzeugt das Verzeichnis verzeichnis
rmdir verzeichnis	(remove dir) löscht das Verzeichnis verzeichnis
rm -R verzeichnis	(remove recursively) löscht das Verzeichnis verzeichnis rekursiv
more datei, less datei	zeigt den Inhalt der Datei datei an
pwd	(print working directory) zeigt den aktuellen Verzeichnispfad an

spezielle Verzeichnisnamen

.	das aktuelle Verzeichnis
ls .	gibt den Inhalt des aktuellen Verzeichnisses aus
..	das übergeordnete Verzeichnis
cd ..	wechselt ins übergeordnete Verzeichnis
~	das Homeverzeichnis des Benutzers
/	das Wurzelverzeichnis

3 Loslegen mit Matlab

Matlab Links

- Webseite der Firma Mathworks:
<http://www.mathworks.de>
- Matlab Tutorials bei Mathworks:
http://www.mathworks.de/academia/student_center/tutorials/

- Einführung in Matlab von Günter Gramlich:
<http://www.hs-ulm.de/users/gramlich/EinfMATLAB.pdf>
- Matlab, eine freundliche Einführung:
<http://www-m3.mathematik.tu-muenchen.de/twiki/pub/M3/Allgemeines/Skripten/matlab.pdf>
- Ein kostenloser Matlab-Clone ist GNU Octave. Dieser ist weitgehend mit Matlab kompatibel. In vielen Linuxdistributionen ist das Programm enthalten, ansonsten kann man es aber herunterladen von:
<http://www.gnu.org/software/octave/>

Starten von Matlab

Aufruf aus einer Linux Kommandozeile (Shell)

\$ matlab mit grafischer Oberfläche

\$ matlab -nodesktop -nosplash ohne grafische Oberfläche

Weitere Optionen erhält Du mit: `matlab -h`

Das Wichtigste ist bekanntlich immer zu wissen, wie man am schnellsten Hilfe bekommt: **Die Online Hilfe**

```
>> help <Befehl>      Matlab Online Hilfe zu Befehl
>> lookfor <Stichwort> Durchsucht die gesamten Hilfen nach Stichwort
>> doc <Befehl>      Ausführliche, verlinkte Online Hilfe
>> demo              Startet einige Matlab-Demos
```

4 Matlab interaktiv

Der folgende Crashkurs soll einen ersten Einblick in Matlab via “Learning by Doing” geben.

Aufgabe 2. *Gib jeweils die hinter dem Prompt >> angegebenen Befehle auf der Matlab Kommandozeile ein und beobachte, was passiert. Schreibe Dir dazu auf dem Arbeitsblatt kurze Kommentare hinter den Befehl.*

Benutze auch die Onlinehilfe um die Bedeutung der Befehle herauszufinden.

Variablen, Vektoren, Matrizen

```
>> a=7          a wird als Skalar (1 × 1-Matrix) interpretiert
>> b=[1,2,3]   nach Komma: neues Element in derselben Zeile, hier also  $b \in \mathbb{R}^{1,3}$ 
>> c=[1+2,3,4]
>> d=[1 1 1]
>> e=[a 7 2*a]
>> f=[1;2;3;4]
>> g=f(3)      weise g das dritte Element von f zu
>> A=[1 2 3;4 5 6]
>> B(3,4)=a    erzeugt eine 3 × 4-Matrix mit a an der Position (3,4)
>> B(1,2)=1
```

```

>> B(4,3)=2
>> h=1:10
>> k=1:2:10
>> x=0:1/3:2
>> B(1:2,2:3)=[1 2;3 4] belegt die Untermatrix  $(B_{i,j})_{\substack{i=1,2 \\ j=2,3}}$  mit den Einträgen  $\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$ 
>> who
>> help who
>> clear a b löscht die Variablen a und b, kontrolliere mit who
>> clear
>> C=[1 2]; Semikolon am Ende unterdrückt die Ausgabe
>> C
>> pi
>> A=eye(3)
>> b=[1:3]
>> B=diag(b)
>> C=diag([1 7 8])
>> D=diag([1;7;8]) geht auch mit Spaltenvektoren
>> E=ones(4) 4× 4-Matrix mit Einsen
>> F=ones(2,3)
>> G=zeros(4)
>> I=[A B;zeros(3) A] Blockmatrix
>> R=rand(4,3)
>> C C ist immer noch gespeichert
>> K=[1;2]+[3;4]*i auch komplexe Zahlen sind möglich,  $K \in \mathbb{C}^{2,1}$ 
>> R.', K.' Transponierte
>> R', K'
>> w(3)=5 Matlab interpretiert w als Zeilenvektor,  $w_3 = 5$ 
>> for i=1:10 Hier ist i ein Laufindex von 1 bis 10, Matlab führt erst
    y(i)=2*i bei end die Befehle in der Schleife aus. Beachte: i = 0 ist
    end hier nicht möglich, da Vektoren in Matlab kein '0.Element' haben!
>> z=2*(1:10) kürzere Alternative, in z und y steht dasselbe

```

Einfache Matrix- und Vektoroperationen

```
>> clear
```

```

>> A=[1 2 3;2 1 0], B=[2 2;1 0;0 1], C=[0 1 0;5 1 3]
>> size(A)
>> [m,n]=size(A)  m= Zeilenzahl, n=Spaltenzahl
>> b=[2 1 3], x=[2;1;3]
>> A*B          Matrixmultiplikation
>> A*C
>> A, C
>> A*C'         $A \cdot C^H$ 
>> diag(A*C')
>> ans         (für "answer") ist die unbenannte Ausgabevariable
>> D=A+C
>> E=A+B       Fehler, da die Dimensionen nicht zusammenpassen!
>> E=A-B'      $E = A - B^H$ 
>> g=A*x
>> g=A*b
>> A,b,B
>> f=b*B       Zeilenvektor mal Matrix
>> C=[1 2 3]'

```

Matrixmanipulationen

```

>> clear; A=[1 2;3 4]
>> A(3,2)=7    Hinzunahme einer dritten Zeile, 7 an Stelle (3,2) der Rest wird mit
                Nullen aufgefüllt.
>> A(1:2,2)    (1:2,2): 1. bis 2. Element der 2. Spalte
>> A(3,1:2)
>> B(3:4,3)=[5;6]
>> C(4:5,4)=A(1:2,2)
>> B(:,3)      3. Spalte von B
>> d=C(5,:)
>> E=[1 2 3;4 5 6;7 8 9;10 11 12]
>> E(1:2:4,3)  Gibt in 2er Schritten die Werte des 1.-4. Elementes der 3. Spalte aus
>> F=[1 2 3 4 5;6 7 8 9 10;11 12 13 14 15]
>> G=F(1:2:3,1:2:5)  Sucht in der 1., 3. und 5. Spalte jeweils vom

```

1. bis 3. Element jedes 2. Element heraus.

```
>> F(:, [1 2 4 5])
>> H=[1 3;9 11]
>> H^(-1)
>> inv(H)
>> H*inv(H)
>> det(H)
>> b=[99 100 101]
>> F(1,1:3)=b
>> A
>> A(1,1:3)=b
```

Das (2, 3)- und das (3, 3)-Element werden neu hinzugefügt.

Grafiken

```
>> x=linspace(0,1,11)      alternativ: x=(0:0.1:1)
```

```
>> y=x.^2, z=sqrt(x),
```

```
>> plot(x,y)
```

```
>> plot(x,z)
```

```
>> clf
```

```
>> plot(x,y)
```

```
>> hold on
```

```
>> plot(x,z)
```

```
>> plot(x,2*z, 'r')
```

```
>> plot(x,y+z, 'g')
```

```
>> hold off
```

```
>> plot(x,y-z, 'k')
```

```
>> more on
```

die Hilfe wird Seitenweise angezeigt (schließen mit q)

```
>> help plot
```

```
>> title('Meine Grafik')
```

```
>> xlabel('x-Achse')
```

```
>> ylabel('y-Achse')
```

```
>> axis([0,20,-5,50])
```

```
>> box on
```

```
>> grid on
```

```

>> clf
>> subplot(3,2,1)
Der Plot hat  $3 \times 2 = 6$  Subplots. Der 1. Subplot wird
angesprochen.

>> plot(x,y)
>> subplot(3,2,2)
>> plot(x,z,'k')
>> subplot(3,2,5)
>> plot(x,z+y,'mo')
>> hold on
>> plot(x,z,'k')
>> subplot(3,2,1)
>> plot(x,z,'k')
>> subplot(3,2,4)
>> title('leer')
>> subplot(3,1,2)
Jetzt gibt es nur noch drei Subplots, einer pro Zeile. Der
2. wird angesprochen.

>> plot(y)
>> orient tall
>> help orient
>> print -deps test1.eps Erzeugt ein eps-File test1.eps dieses Plots.
>> print -depsc test2.eps

>> help print
>> !ls
Matlab gibt alles was hinter einem '!' steht an Unix weiter.
Für ls siehe oben. (nicht unter Windows!)

>> !gv test2.eps

```

... und hiermit startest Du viele schöne Matlab-Demos:

```

>> demo
>> demo matlab graphics

```