MATLAB Mini-Einführung

Autorenkollektiv der PPM/Thomas Slawig

Projektgruppe Praktische Mathematik
Institut für Mathematik
TU Berlin

4. November 2002

Inhaltsverzeichnis

1 Preliminarien und Start

Matlab ist die Abkürzung von *Matrix Laboratory* und die Bezeichnung für eine kommerzielle[1] mathematische Software, die besonders geeignet ist für Vektor- und Matrixoperationen.

1.1 Was ist eine Matrix?

Eine Matrix ist ein zweidimensionales, rechteckiges Zahlenschema der Form

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$$

mit einer festen Anzahl von Zeilen (hier zwei) und Spalten (hier drei). Die Elemente der Matrix A werden mit A_{ij} bezeichnet, dabei bezeichnet der erste Index (hier i) die Zeile und der zweite (hier j) die Spalte. In mathematischer Schreibweise gilt $A \in \mathbb{R}^{m \times n}$, wenn die Matrix A aus reellen Zahlen besteht und genau m Zeilen und n Spalten hat. Man spricht von einer $(m \times n)$-Matrix.

Vektoren sind damit nur noch spezielle Matrizen: Ein Zeilenvektor der Länge n ist eine $(1 \times n)$-Matrix, ein gleich langer Spaltenvektor eine $(n \times 1)$-Matrix. Selbst eine einzelne reelle Zahl (ein Skalar) wird so zu einer (1×1)-Matrix.

1.2 Matlab starten und beenden

Durch die Eingabe des Kommandos

```
matlab
```

wird Matlab gestartet. Es erscheint eine Eingabeaufforderung (oder Prompt) in der Form

```
>>
```

an dem Befehle eingegeben werden können, die dann sofort ausgeführt (interpretiert) werden.

Mit dem Befehl `exit` oder `quit` wird Matlab beendet.

1.3 Eigenschaften von Matlab

Matlab bietet

- eine Vielzahl von elementaren mathematischen Funktionen,
- gute numerische Algorithmen für sehr viele spezielle mathematische Probleme,
- Kontrollstrukturen (Abfragen und Iterationen) so daß eine Art Programmierung möglich ist,
- eine komfortable und umfangreiche Grafik, so dass Matlab auch zur Visualisierung von mit anderen Programmen erzeugten Ergebnissen dienen kann,

[1] Matlab ist ein Markenzeichen der Firma The Mathworks Inc., USA

- weitere Funktionen zu bestimmten Themen oder Aufgaben (z.B. Statistik), die in sog. Toolboxen zusammengefasst sind.

Da jedes eingebene Kommando interpretiert wird, ist Matlab ein sog. *Interpreter* und ein damit geschriebenes Programm ist langsamer als ein in Maschinensprache übersetztes, z.B. in `C`, `C++` oder `Fortran` geschriebenes Programm.

1.4 Demos und Hilfe

Mit der Eingabe

```
>> demo
```

startet eine Demonstration von Matlab, mit

>> `help` *Kommando*

gibt es Hinweise zu dem entsprechenden Kommando. Der Befehl

>> `lookfor` *keyword*

gibt aus allen Matlab-Dateien die Sätze aus, die den Begriff *keyword* enthalten. Mit

```
>>  helpdesk
```

wird ein Browser (Netscape) mit einer komfortablen Hilfe gestartet.[2] Dort gibt es u.a. die Möglichkeit Informationen über Matlab-Funktionen nach Thema oder nach Index zu suchen.

2 Grundrechenarten und elementare Funktionen

Neben den Standardoperatoren `+,-,*,/,^, sqrt` (für Wurzel) gibt es die in Tabelle 1 aufgelisteten elementaren Funktionen.

3 Variablen und vordefinierte Konstanten

Variablen sind mit Buchstaben oder Buchstabenfolgen bezeichnete Objekte, deren Wert gesetzt und verändert werden kann. Matlab unterscheidet Groß- und Kleinschreibung.

Variablen müssen in Matlab nicht extra deklariert werden, wie sonst bei höheren Programmiersprachen üblich. Mit

```
>> x=1
```

wird der Variable X ein Wert zugewiesen und sie kann von da ab benutzt werden, z.B.

```
>> y=x+1
```

[2]Die angezeigten HTML-Dateien liegen local auf einem Server und müssen also nicht von weither übertragen werden.

Trigonometric			
sin	Sine	sinh	Hyperbolic sine
asin	Inverse sine	asinh	Inverse hyperbolic sine
cos	Cosine	cosh	Hyperbolic cosine
acos	Inverse cosine	acosh	Inverse hyperbolic cosine
tan	Tangent	tanh	Hyperbolic tangent
atan	Inverse tangent	atan2	Four quadrant inverse tangent
atanh	Inverse hyperbolic tangent	sec	Secant
sech	Hyperbolic secant	asec	Inverse secant
asech	Inverse hyperbolic secant	csc	Cosecant
csch	Hyperbolic cosecant	acsc	Inverse cosecant
acsch	Inverse hyperbolic cosecant	cot	Cotangent
coth	Hyperbolic cotangent	acot	Inverse cotangent
acoth	Inverse hyperbolic cotangent		
Exponential			
exp	Exponential	log	Natural logarithm
log10	Common (base 10) logarithm	log2	Base 2 logarithm and dissect floating point number
pow2	Base 2 power and scale floating point number	nextpow2	Next higher power of 2
sqrt	Square root		
Complex			
abs	Absolute value	angle	Phase angle
conj	Complex conjugate	imag	Complex imaginary part
real	Complex real part	unwrap	Unwrap phase angle
isreal	True for real array	cplxpair	Sort numbers into complex conjugate pairs
Rounding and remainder			
ceil	Round towards plus infinity	fix	Round towards zero
floor	Round towards minus infinity	round	Round towards nearest integer
mod	Modulus (signed remainder after division)	rem	Remainder after division
		sign	Signum

Tabelle 1: Elementare Funktionen

Ein Semikolon `;` hinter einem Befehl bewirkt, dass das Ergebnis *nicht* auf dem Bildschirm ausgegeben wird. Soll der Wert der Variable x ausgegeben werden, so gibt man

```
>>
```

ein. Man kann mehrere, jeweils durch ein Semikolon getrennte Befehle in einer Zeile schreiben.

Matlab hat einige vordefinierte Konstanten, z.B. `pi` für π und `i` für die imaginäre Einheit. Man kann solche Konstanten neu definieren, also z.B.

```
>> i=1
```

und `i` dann als Zählvariable verwenden.

4 Vektoren und Matrizen

Vektoren und Matrizen werden in Matlab in eckigen Klammern angegeben. Dabei steht

- zwischen den einzelnen Elementen in einer Zeile ein Leerzeichen oder Komma, z.B. für einen Zeilenvektor

```
>> x = [1/2 pi sqrt(4)-1]
```

- zwischen den einzelnen Zeilen wahlweise <Enter> oder ein Semikolon, z.B. für einen Spaltenvektor

```
>> x = [1/2; pi; sqrt(4)-1]
```

Die Dimension eines Vektors oder einer Matrix muß in Matlab nicht extra angegeben werden.

Auf Elemente eines Vektors oder einer Matrix kann über Indizes zugegriffen werden, die in *runden* Klammern stehen: So ergibt im obigen Beispiel

```
>> x(2)
```

die Ausgabe 3.1416. Auch ist es möglich, den Vektor x wie folgt zu erweitern:

```
>> x(5) = 1
```

Über das vierte Element wurde hier keine Aussage gemacht. Deshalb bekommt es bei Matlab automatisch den Wert Null. Die Anweisung

```
>> x(4) = []
```

weist dem vierten Element einen leeren Vektor zu. Damit wird dieses Element eliminiert und die Dimension von x um eins runtergesetzt.

Vektoren können wie einzelne Elemente zu größeren Vektoren zusammengefaßt werden. Aus

```
>>  a = [1 2]
>>  b = [3 4]
```

kann der Vektor

```
>>  c = [a b]
    c =
         1     2     3     4
```

gebildet werden. Analog funktioniert das Ganze mit Matrizen:

```
>>  A  =  [ 1  2  3
            4  4  4
            5  6  9 ]
```

Das geht auch in der kompakteren Schreibweise:

```
>>  A  =  [ 1  2  3  ;  4  4  4  ;  5  6  9]
```

Mit

```
>>  A(2,3)
```

fragt man das Element A_{23} ab. Aus

```
>>  A = [1 2 ; 3 4]
>>  B = [1 1 ; 1 1]
```

sollen neue Matrizen gebildet werden. Dabei können die Elemente nebeneinander

```
>>  C = [A B]
C =
      1     2     1     1
      3     4     1     1
```

oder untereinander angeordnet werden.

```
>>  D = [A; B]
D =
     1     2
     3     4
     1     1
     1     1
```

4.1 Intervalldarstellung

Man kann Vektoren auch bilden, indem man das Intervall und den Zuwachs (increment) angibt. Mit der folgenden Anweisung wird ein Vektor mit den Elementen $\{0, \frac{1}{3}, \frac{2}{3}, 1\}$ erzeugt, gemäß der Schreibweise: `Startwert:Schrittweite:Endwert`. Solange die Schrittweite 1 ist, kann die Schreibweise auch abgekürzt werden zu: `Startwert:Endwert`.

```
>>  y = 0:1/3:1
y =
      0 0.3333 0.6667 1
```

4.2 Teilvektoren und -Matrizen

Einzelne Elemente eines Vektors x können mit der Indexschreibweise `x(i)` angesprochen werden. Nun kann i aber selbst auch ein Vektor sein.

```
>>  i = 2:5
>>  y = x(i)
```

entspricht

```
>>  y = x(2:5)
```

und liefert einen Vektor, bestehend aus der zweiten bis fünften Komponente von x. Analog für Matrizen:

```
>>  i = 1:2
>>  E = C(i,1)
```

Die Matrix E enthält dann die Elemente der ersten Spalte: $C(i(1),1),...,C(i(n),1)$. Die Anweisung

```
>>  E = C(1:2,1)
```

ist ebenfalls möglich. Mit Hilfe des Operators : können alle Elemente einer Spalte oder Zeile angesprochen werden.

```
>>  E = C(:,1)
```

Weitere Befehle:

`linspace(x1,x2,n)`	erzeugt einen Vektor mit n Elementen in gleichen Abständen zwischen $x1$ und $x2$ (*linearly equally spaced*)
`length(x)`	Länge des Vektors x
`size(A)`	Dimensionen der Matrix A

4.3 Rechnen mit Vektoren und Matrizen

Die Grundrechenarten können auf Vektoren und Matrizen angewandt werden, wenn sie Sinn machen. Zwei Vektoren oder Matrizen gleicher Dimensionen können einfach mit

```
>>  X+Y
```

addiert oder subtrahiert werden. Mit

```
>>  X+1
```

wird zu der beliebig dimensionierten Matrix X der konstante Vektor mit den Elementen 1 addiert. Bei Multiplikation, Division, Potenzieren, und Wurzelziehen muss man dem Operator einen Punkt voranstellen, wenn er elementweise wirken soll. So berechnet

```
>>  Z=X.^Y
```

für zwei gleichdimensionierte Matrizen X und Y eine Matrix mit den Elementen $Z_{ij} = X_{ij}^{Y_{ij}}$.

4.4 Funktionen auf Vektoren und Matrizen

Matlabs Funktionen können auch auf Vektoren und Matrizen angewandt werden. Die Berechnung erfolgt elementweise. Ist X ein Vektor oder Matrix mit Dimension $(m \times n)$, so ist

```
>>  Y = sin(X)
```

eine Matrix derselben Dimension wie X mit den Komponenten $Y_{ij} = \sin(X_{ij})$. Weitere elementare Funktionen auf Vektoren und Matrizen sind

`ones(n)`	erzeugt eine $(n \times n)$-Matrix mit Einsen
`ones(m,n)`	erzeugt eine $(m \times n)$-Matrix mit Einsen
`zeros(n)`	erzeugt eine $(n \times n)$-Matrix mit Nullen
`zeros(m,n)`	erzeugt eine $(m \times n)$-Matrix mit Nullen
`eye(n)`	n-dimensionale *Einheitsmatrix*: Einsen auf der Diagonale, sonst Nullen
`min(x)`	findet das kleinste Element eines Vektors (– jeder Spalte einer Matrix)
`max(x)`	findet das größte Element eines Vektors (– jeder Spalte einer Matrix)
`sum(x)`	summiert die Elemente eines Vektors auf
`sort(x)`	sortiert einen Vektor in aufsteigender Reihenfolge
`mean(x)`	bildet den Mittelwert eines Vektors (oder der Vektoren einer Matrix)
`char(x)`	konvertiert numerische Vektoren entsprechend ASCII in Buchstaben

5 Skriptdateien (m-Files)

Matlab führt einzelne nach dem Prompt eingegebene Kommandos aus. Bequemer ist es, eine Folge von Befehlen in einer Datei zu speichern und abarbeiten zu lassen, ohne sie einzeln eingeben zu müssen. Solche Dateien werden Skriptdateien (m-Files) genannt. Die im Editor erstellte Datei erhält dazu die Endung `.m`. Existiert eine Datei `script.m`, so werden mit

```
>> script
```

die in ihr stehenden Matlab-Befehle abgearbeitet. Danach verbleiben die in der Datei verwendeten Variablen im Arbeitsspeicher. Kommentare werden in Matlab mit `%` angezeigt, nach einem `%` ist der Rest der Zeile ein Kommentar (wie in LaTeX).

6 Von Datei lesen, auf Datei schreiben

Daten von Datei einlesen ist in Matlab denkbar einfach, vor allem, wenn die Datei im ASCII-Format vorliegt und die Daten in Form von Zeilen und Spalten enthält. Mit

```
>>  load T.dat
```

wird der Inhalt der Datei `T.dat` einer Variablen (Matrix) mit dem Namen `T` zugewiesen. Durch Eingabe des Variablennames – ohne Semikolon dahinter – bekommt man eine Terminalausgabe der Variablen. Um diese Matrix wieder zu speichern (unter einem anderen Namen, z.B. `A`), wird das Kommando `save` verwendet.

```
>>  save A.dat T -ascii
```

Die Datei soll ein lesbares Textfile sein, deshalb ist die Angabe `A.dat` und die Option `-ascii` erforderlich. Ohne diese Angaben wird der Inhalt der Variablen `T` in die Datei `tTemp.mat` geschrieben und zwar im Matlab-Binärformat.

```
>>  save T
```

Weitere Informationen zu den Befehlen erhält man wieder mit `help` *befehlsname*.

Die bisher erstellten Variablen werden im Matlab Arbeitsspeicher (*Workspace*) gespeichert. Der Befehl

```
>>  whos
```

listet alle Variablen im aktuellen Arbeitsspeicher auf.

```
>>  clear
```

löscht alle Variablen im aktuellen Workspace.

7 Unix-Kommandos

Matlab *kennt* einige Unix-Kommandos, wie zum Beispiel

`ls, pwd, mkdir, cd, dir,`

d.h. es gibt Matlab m-files, die den gleichen Namen und annähernd die gleiche Funktion haben wie entsprechende Unix-Kommandos. Zusätzlich ist es möglich, alle weiteren Unix-Kommandos durch ein vorangestelltes `!` an eine Shell (Unix-Kommandointerpreter, xterm) weiterzureichen.

8 Grafik

Matlab bietet viele komfortable Grafikfunktionen. Diese Liste hat nicht den Anspruch *alles* darzustellen, weiteres findet sich in der Matlab-Hilfe.

8.1 Erstellen von Plots

Zweidimensionale Darstellung

`plot(y)`	trägt die Elemente des Vektors y über den jeweiligen Index von y auf
`plot(x,y)`	trägt Vektor y über Vektor x auf
`plot(t,y1,t,y2,t,y3)`	drei Graphen werden mit einem Aufruf gezeichnet
`loglog( )`	doppeltlogarithmische Auftragung der Argumente (Argumente wie beim Befehl `plot`)

Beispiel

```
>>  t = 0:pi/100:2*pi;       % Vektor t definieren
>>  y = sin(t);              % Vektor y erzeugen
>>  plot(t,y);
```

Dreidimensionale Darstellung

`plot3( )`	plottet Linien und Punkte in 3-D (Argumente wie beim Befehl `plot`)
`[X,Y]=meshgrid(x,y)`	generiert zwei Matrizen die für die 3D-Darstellung erforderlich sind
`mesh()`	legt ein Gitter über den Plot
`surf()`	füllt die Oberfläche des Plots aus (zusätzlich 3D)
`quiver(x,y)`	zeichnet Pfeile mit den Komponenten x,y
`contour(x)`	erzeugt die Höhenlinien zu x

Hinweis: Der Befehl `help graph3d` zeigt euch mehrere interessante Befehle auf einem Blick.

Beispiel (für quiver)

```
>>  x = [0:.2:1];            % Vektor x definieren
>>  y=x.^2;                  % Vektor y erzeugen
>>  u=gradient(x);           % bildet den Gradienten von x
>>  v=gradient(y);
>>  s=.5;                    % Normierung der Pfeile
>>  quiver(x,y,u,v,s);
>>  hold on                  % Die folgenden Befehle werden in den
                             % aktuellen Graph hinzugeplottet

>>  y = .1+x.^2;             % Erweiterung
>>  v=gradient(y);           % bildet den Gradienten von y
>>  s=.2;
>>  quiver(x,y,u,v,s);
>>  hold off;
```

8.2 Einstellungsmöglichkeiten

Achseneinstellungen

`axis([xmin xmax ymin ymax])`	Skalierung der Achsen
`grid on/off`	Gitterlinien werden ein bzw. ausgeschaltet
`hold on/off`	Graphikfenster bleibt für weitere Plots

Graphenbeschriftung

`xlabel('Name')`	x-Achsenbeschriftung
`ylabel('Name')`	y-Achsenbeschriftung
`title('Titel')`	Titel
`text(x,y,'Text')`	beliebiger Text wird an die Stelle x,y im Graph geschrieben
`gtext('Text')`	beliebiger Text wird per Maus an gewünschte Stelle gesetzt
`legend('Text1','Text2',...)`	Legende für alle geplotteten Graphen

Beispiel

```
>>  t = 0:pi/100:2*pi;
>>  y = sin(t);
>>  plot(t,y);
>>  xlabel('0 \leq t \leq \pi')        % 0<= t <=pi
>>  ylabel('y')
>>  title('Sinuskurve')
>>  text(4,0.2,'Matlab macht Spass!')
>>  legend('sin(t)')
```

Weitere Einstellungen

`shading interp`	interpoliert die diskreten Werte
`colorbar`	zeichnet eine Farbskala
`colormap()`	Auswahl eines Farbschemas(siehe helpdesk colormap)
`pcolor(Z)`	farbiges plot wo die Werte der Matrix die Farbe bestimmen
`view(`α, β`)`	setzt den Blickpunkt auf die Graphik fest
`pause(5)`	das momentane Bild wird 5 Sekunden gehalten, bevor das Programm weiterarbeitet
`pause`	das Programm erwartet eine enter-Eingabe

8.3 Abspeichern und Ausdrucken

Nun kann man die Graphik natürlich auch noch ausdrucken oder auf eine Datei schreiben. Alle Optionen finden sich wie immer in der Hilfe.

Die allgemeine Form ist: `print` -*devicetype* -*option filename*

```
>>  print -dps2 filename            % Postscript(s/w)
>>  print -dpsc2 filename           % farbiges Postscript
>>  print -deps2 filename           % encapsulated Postscript
>>  print -depsc2 filename          % farbiges EPS
>>  print -Pdruckername             % ausdrucken
```

Dies alles geht auch im Graphikfenster. Im Graphikfenster wählt man im Menü $\boxed{\text{File}}$ die Option $\boxed{\text{Print}}$ aus. Es poppt ein neues Fenster auf. Dort wählt man dann entweder $\boxed{\text{Print}}$ (ausdrucken) oder $\boxed{\text{Save}}$ (schreiben auf Datei) aus. Auch hier müssen die obenstehenden Optionen eingetragen werden.