Kapitel 4

Einführung in MATLAB

4.1 Allgemeines

MATLAB ist eine kommerzielle mathematische Software zur Lösung mathematischer Probleme und zur graphischen Darstellung der Ergebnisse. Die Verfahren in MATLAB beruhen auf Matrizen (MATrix LABoratory).

MATLAB ist leider nicht ganz billig. Im Computer–Pool ist eine Classroom–Lizenz (25 Stück) von MATLAB 7.3 (R2006b) vorhanden. Im Moment kann das Programm mit dem Skript

`matlab.sh`

gestartet werden, welches von der gleichen Homepage geladen werden kann, auf der die Übungsblätter zu finden sind (Aufruf `sh matlab.sh`).

Zur Einarbeitung in MATLAB gibt es viele Bücher, siehe www.amazon.de. Das Buch [DS04] ist so eine Art Klassiker, der einen kurzen und knappen Überblick gibt (man muss wissen, wonach man suchen soll). Eine Uraltversion von [DS04] ist im Internet (siehe Homepage, auf der die Übungen stehen) verfügbar. Weitere frei verfügbare Beschreibungen findet man auf der gleichen Homepage und im Internet. Diese Dokumentationen beruhen zwar auf älteren Versionen von MATLAB, sind aber für diese Vorlesung vollkommen ausreichend. Es gibt eine umfangreiche und gute Hilfe innerhalb von MATLAB, Aufruf mit `help`.

Man programmiert in MATLAB mit einer plattformunabhängigen Programmiersprache, die auf der jeweiligen Maschine interpretiert wird. Durch den einfachen, mathematisch orientierten Syntax der MATLAB–Skriptsprache und durch umfangreiche vorhandene Funktionsbibliotheken ist die Erstellung von Programmen wesentlich einfacher möglich als beispielsweise unter C. Allerdings sind MATLAB–Programme im allgemeinen bedeutend langsamer als C–Programme.

Man kann sein Programm direkt in das MATLAB–Befehlfenster eintippen. Sinnvoller ist es jedoch, es in eine seperate Datei zu tun und diese vom MATLAB–Befehlfenster aus aufzurufen. *Vorlesung: an Summe der ersten 100 Zahlen demonstrieren.* Mit dem Befehl `edit` wird ein Editor geöffnet, in dem man die Datei erstellen kann. MATLAB–Befehlsdateien besitzen die Endung `.m`, (M–Files). Mit dem Befehl `what` kann man sich die im gegenwärtigen Verzeichnis vorhandenen M–Files ansehen. Sie werden ausgeführt, indem sie im MATLAB–Befehlfenster einfach aufgerufen werden (die benötigten Parameter müssen natürlich übergeben werden). Weitere wichtige allgemeine MATLAB–Befehle sind `ls, cd, pwd`. Sie haben die gleiche Bedeutung wie in LINUX. Des weiteren sind die Befehle

`clear;` löscht alle Variablen
`clf;` löscht alle Bildfenster

`who;` zeigt alle Variablen an

wichtig, damit bei einem wiederholten Starten von Programmen nicht alte Belegungen die Ergebnisse verfälschen. Die Ausgabe von Text erfolgt mit `disp`. Die Formatierung mit `format`.

Die Nutzung von MATLAB ist an vielen Hochschulen Standard im Rahmen von Vorlesungen, die sich mit numerischen Verfahren beschäftigen. Hier werden nur die wichtigsten Befehle vorgestellt. Ansonsten gilt, was für jede Programmiersprache gilt: *Learning by doing.*

4.2 Bemerkungen zu Vektoren und Matrizen

Vektoren sind aus der Schule bekannt. Man unterscheidet

$$\mathbf{a} = (a_1, \ldots, a_n),$$

einen n–dimensionalen Zeilenvektor und

$$\mathbf{b} = \begin{pmatrix} b_1 \\ \vdots \\ b_n \end{pmatrix},$$

einen n–dimensionalen Spaltenvektor. Die Anzahl der Komponenten eines Vektors nennt man Dimension (hier n, in der Schule im allgemeinen $n \in \{2, 3\}$).

Wandelt man einen Zeilenvektor in einen Spaltenvektor mit den gleichen Einträgen um (oder Spalten– in Zeilenvektor), so nennt man diese Operation transponieren. Der transponierte Vektor des obigen Zeilenvektors ist

$$\mathbf{a}^T = \begin{pmatrix} a_1 \\ \vdots \\ a_n \end{pmatrix}.$$

Das Skalarprodukt zweier Spaltenvektoren ist aus der Schule für $n = 3$ bekannt. Für zwei n–dimensionale Vektoren ist es

$$\mathbf{a} \cdot \mathbf{b} = (\mathbf{a}, \mathbf{b}) = \mathbf{a}^T \mathbf{b} \sum_{i=1}^{n} a_i b_i.$$

Die Norm oder Länge eines Vektors ist

$$\|\mathbf{a}\| = (\mathbf{a} \cdot \mathbf{a})^{1/2} = \left(\sum_{i=1}^{n} a_i^2 \right)^{1/2}.$$

Matrizen und ihre tiefere Bedeutung sowie ihre Eigenschaften werden am Ende von Lineare Algebra I behandelt. Hier ist es nur wichtig, dass es zwei–dimensionale Felder sind, mit m Zeilen und n Zeilen:

$$A = \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{pmatrix}.$$

In diesem Sinne sind n–dimensionale Zeilenvektoren $1 \times n$ Matrizen und m–dimensionale Spaltenvektoren $m \times 1$ Matrizen.

4.3 Matrizen

MATLAB rechnet mit Matrizen. Dabei ist zum Beispiel eine Zahl eine 1×1–Matrix und ein Spaltenvektor eine $n \times 1$–Matrix. Die Dimensionskontrolle wird in MATLAB streng durchgeführt. So ist es zum Beispiel nicht möglich, einen Zeilen– und einen Spaltenvektor der gleichen Länge zu addieren. Die Indizierung von Matrizen beginnt in MATLAB mit 1.

MATLAB rechnet auch mit komplexen Zahlen. Wichtige Konstanten sind:

- `pi`
- `i`, `j`, imaginäre Einheit.

Die Eingabe beziehungsweise Erzeugung von Matrizen kann auf verschieden Art und Weisen erfolgen:

- Eingabe der Matrixeinträge,
- Laden aus externen Dateien,
- Erzeugen mit gegebenen Funktionen,
- Erzeugen mit eigenen Funktionen.

Beispiel 4.1 Die Matrix

$$A = \begin{pmatrix} 2 & 4 & 7 \\ -5 & 4 & 2 \end{pmatrix}$$

kann wie folgt eingegeben werden

```
A = [2 4 7; -5, 4, 2]
```

Man kann auch jedes Element einzeln angeben:

```
A(1,1) = 2
A(1,2) = 4
A(1,3) = 7
A(2,1) = -5
A(2,2) = 4
A(2,3) = 2
```

Die Einheitsmatrix der Dimension $n \times n$ erhält man mit

```
B = eye(n)
```

wobei `n` vorher mit einer positiven ganzen Zahl belegt sein muss. Analog erhält man eine Matrix mit Nullen durch

```
C = zeros(n)
```

Eine $(m \times n)$–Matrix mit zufälligen Einträgen erhält man mit

```
D = rand(m,n)
```

Will man die Ausgabe der Matrizen auf dem Bildschirm unterdrücken, so beendet man die Befehle mit einem Semikolon. □

Wichtige Operationen mit Matrizen:

- Die transponierte Matrix A^T einer gegebenen Matrix `A` erhält man mit

  ```
  B = A'
  ```

  Man kann die transponierte Matrix auch auf dem Speicherplatz der ursprünglichen Matrix speichern

  ```
  A = A'
  ```

- Die Dimension von `A` erhält man mit

  ```
  [m,n] = size(A)
  ```

  Hierbei ist `m` die Anzahl der Zeilen und `n` die Anzahl der Spalten.

- Die Teilmatrix mit den Zeilenindizes $i1, \ldots, i2$ und den Spaltenindizes $j1, \ldots, j2$ einer Matrix `A` erhält man mit

  ```
  B = A(i1:i2,j1:j2)
  ```

Wird der Doppelpunktoperator ohne vorderes Argument gebraucht, so wird mit dem ersten Index begonnen; ohne hinteres Argument, wird mit dem letzten Index aufgehört. So erhält man die erste Zeile von `A` durch

```
B = A(1,:)
```

- Addition zweier Matrizen `A` und `B`, Subtraktion sowie Multiplikation werden mit den üblichen Symbolen bezeichnet

  ```
  C = A+B      C = A-B      C = A*B
  ```

- Multiplikation, Division und Potenzierung einer Matrix `A` mit einem Skalar a ((1×1)–Matrix) werden mit den üblichen Symbolen bezeichnet

  ```
  B = a*A      B = A/a      B=A^a
  ```

- Elementweise Multiplikation und Division werden wie folgt durchgeführt

  ```
  C = A.*B      C = A./B
  ```

  Beispiel:

$$\mathtt{A} = (1, 5), \quad \mathtt{B} = \begin{pmatrix} 3 \\ -2 \end{pmatrix}, \qquad \mathtt{A} * \mathtt{B} = -7, \quad \mathtt{A}. * \mathtt{B}' = (3, -10)$$

Die folgende Liste enthält wichtige Befehle, die man für Matrizen in MATLAB zur Verfügung hat. Die Einfachheit dieser Befehle sollte nicht darüber hinwegtäuschen, dass innerhalb von MATLAB zum Teil komplizierte Verfahren zur Berechnung der Ergebnisse ablaufen (siehe Vorlesung Praktische Mathematik). Vergleichbare Befehle stehen zum Beispiel in C nicht zur Verfügung. Daraus erklärt sich die Einfachheit, mit der man Programme in MATLAB erstellen kann. Für die angegebenen Befehle gibt es teilweise alternative Aufrufe, siehe MATLAB–Hilfe.

- der Rang einer $n \times n$–Matrix `A`:

  ```
  r = rank(A)
  ```

- die Inverse einer regulären $n \times n$–Matrix `A`:

  ```
  B = inv(A)
  ```

- die Determinante einer $n \times n$–Matrix `A`:

  ```
  d = det(A)
  ```

- die Spektralnorm einer $m \times n$–Matrix `A`:

  ```
  d = norm(A)
  ```

  Andere Normen können ebenfalls berechnet werden, siehe MATLAB–Hilfe. Ist `A` ein Vektor, dann ist die Spektralnorm die Euklidische Vektornorm.

- die Eigenwerte und Eigenvektoren einer $n \times n$–Matrix `A`:

  ```
  [u,v] = eig(A);
  ```

  Dabei enthält `v` eine Diagonalmatrix mit den Eigenwerten und die Matrix `u` enthält die zugehörigen Eigenvektoren.

- die Lösung eines linearen Gleichungssystems `Ax = b` mit einer regulären $n \times n$–Matrix `A` erhält man mit

  ```
  x = A\b
  ```

4.4 Wichtige Funktionen in MATLAB

Befehl	Bedeutung
`atan`	Arcus–Tangens
`cos`	Kosinus
`exp`	Exponentialfunktion
`log`	Logarithmus naturalis ln !
`log10`	Logarithmus zur Basis 10

`sin`	Sinus
`sqrt`	Wurzel
`tan`	Tangens

Für Einzelheiten, zum Beispiel was bei matrixwertigen Argumenten passiert, siehe MATLAB–Hilfe. Weitere Funktionen findet man ebenso in der MATLAB–Hilfe.

4.5 Ablaufkontrolle

Die Ablaufkontrolle beinhaltet Alternativen und Zyklen.

Die Alternative wird wie folgt programmiert:

```
if expr
   sequenz
elseif expr
   sequenz
else
   sequenz
end
```

Dabei gibt *expr* einen Wahrheitswert (true oder false) zurück. Folgende Relationen und wichtige logische Operatoren stellt MATLAB zum Vergleich von Ausdrücken zur Verfügung:

Befehl	Bedeutung
`<`	kleiner
`<=`	kleiner oder gleich
`>`	größer
`>=`	größer oder gleich
`==`	gleich
`~=`	ungleich
`&&`	logisches und
`||`	logisches oder
`~`	logisches nicht
`xor`	exklusives oder (entweder oder)

Soll zum Beispiel kontrolliert werden, ob eine Matrix ein Zeilen– oder Spaltenvektor ist, so kann man das mit

```
[m,n] = size(a)
if (m==1) || (n==1)
```

tun.

Innerhalb der `if`–Anweisung sind mehr als eine `elseif`–Abfrage möglich. Eine Alternative für Mehrfachverzweigungen bildet die `switch`–Anweisung:

```
switch switch expr
   case expr,
      sequenz
   case expr,
      sequenz
   case expr,
      sequenz
```

```
   otherwise,
      sequenz
end
```

Hier wird der Ausdruck nach dem `switch`–Befehl ausgewertet und dann der entsprechende Teil der Anweisung abgearbeitet, bei welcher der Ausdruck hinter dem `case`–Befehl mit der Auswertung des `switch`–Befehls übereinstimmt. Ein Beispiel gibt es in den Übungen.

Mit einer `for`–Schleife wird eine vorgegebene Anzahl von Zyklen durchlaufen, etwa

```
for i=1:100
   sequenz
end
```

Eine `while`–Schleife wiederholt eine Sequenz so oft, bis ein logisches Abbruchkriterium erfüllt ist, etwa

```
i=1;
while i<100
   i= input('i ');
end
```

Diese Schleife wird erst abgebrochen, wenn eine Zahl $i \geq 100$ eingegeben wird.

Sowohl die `for`– als auch die `while`–Schleife sind abweisend. Ein vorzeitiges Verlassen eine Schleife ist mit dem `break`–Befehl möglich.

4.6 Graphik

Eine Stärke von MATLAB ist die Graphik, mit der man sich schnell und unkompliziert die berechneten Ergebnisse ansehen kann. Wie generell in MATLAB, werden bei der Graphik Daten gezeichnet, die in Vektoren und Matrizen gespeichert sind.

Zweidimensionale Graphiken erhält man mit dem `plot`–Befehl:

```
for i=1:101
   x(i) = (i-1)/100;
   y(i) = x(i)^3-0.5;
end
plot(x,y)
```

Damit wird die Funktion $x^3 - 0.5$ im Intervall $[0, 1]$ gezeichnet. Übergibt man nur ein Argument an `plot`, dann sind die Werte auf der x–Achse die Indizes der Vektoreinträge. Für verfügbare Linienarten und –farben sei auf `help plot` sowie auf die Literatur verwiesen. Von der Hilfe zu `plot` wird man dann auch zu weiteren Befehlen geführt, mit denen man eine Graphik verschönern kann, wie `legend, xlabel, axis`. Man kann die Graphiken auch interaktiv editieren.

Die graphische Darstellung von Flächen über der Ebene geht mit dem `mesh`–Befehl:

```
mesh(x,y,Z)
```

Dabei sind `x,y` die Vektoren mit den x– und y–Koordinaten und in der Matrix `Z` stehen die zugehörigen Funktionswerte: $Z(i, j)$ ist der Funktionswert im Punkt $(x(i), y(j))$.

Will man in eine vorhandene Graphik weitere Bildelemente einfügen, dann nutzt man den Befehl `hold`:

```
hold on
```

Damit werden die vorhandenen Bildelemente nicht gelöscht.