Kurzeinführung Matlab/Octave

Matlab ist eine „command-line“ Sprache, d. h. eingegebene Befehle werden, sofern sie syntaktisch richtig sind, unmittelbar ausgeführt.

Matlab kennt im wesentlichen nur eine Art von Objekten, nämlich Matrizen. Eine Zahl ist also eine 1×1 Matrix. Matlab unterscheidet bei Zahlenwerten nur zwischen reell und komplex. Dies bedeutet, dass jede Zahl in double precision Format abgespeichert ist, unabhängig davon, ob es sich um typische ganzzahlige Größen, wie Laufvariable, handelt, oder nicht.

Matlab interpretiert die üblichen *arithmetischen Verknüpfungssymbole* im Kontext mit Matrizen.

`C = A*B` wird in Matlab daher als Matrixprodukt von A und B interpretiert. Falls A und B nicht die geeignete Dimensionierung für die Multiplikation besitzen, erfolgt eine diesbezügliche Fehlermeldung, und der Befehl wird ignoriert.

Wird ein Befehl durch einen Strichpunkt `;` abgeschlossen, so wird der Befehl ausgeführt, aber das Resultat nicht am Schirm angezeigt. So ergibt die Anweisung

```
>> C = A * B;
```

das Produkt von AB und speichert das Resultat auf C.

Die Anweisung

```
>> C = A * B
```

ohne Strichpunkt (bzw. mit Beistrich), speichert ebenfalls AB auf C ab, zusätzlich wird aber das Resultat, also C am Schirm angezeigt.

Ausgabe der Zahlen am Schirm: Obwohl Matlab intern mit Maschinengenauigkeit arbeitet, erfolgt die Darstellung von Zahlen nur gerundet. So liefert der Befehl

```
>> sqrt(2)
ans =
    1.4142
```

Fügt man jedoch vorher den Befehl `format long` ein, so ergibt sich

```
>> format long
>> sqrt(2)
ans =
   1.41421356237310
```

Es können auch mehrere Befehle in einer Zeile eingegeben werden.

```
a1 = b + c; a2 = norm( a1);
```

Matlab hat eine `help` Funktion, mit der alle Matlab-Befehle abgefragt werden können. So liefert etwa `help det` *Hilfe* über den Determinantenbefehl.

Mit dem Befehl

```
diary filename
```

wird die Schirmausgabe gleichzeitig im File mit Namen *filename* abgespeichert. Dies kann nützlich sein, wenn die Resultate später weiter benutzt werden. Der Befehl

```
diary off
```

stellt die Speicherung ab.

Matlab verfügt über eine Reihe *integrierter mathematischer Befehle.* `>> b = det(a);` liefert die Determinante. `>> c = chol( a);` liefert die Choleskyzerlegung von a. Matlab unterscheidet übrigens zwischen Groß- und Kleinbuchstaben, also ist a von A verschieden.

Man kann auch seine eigenen *Matlab-Routinen* schreiben. Hierzu gibt es zwei Möglichkeiten.

1. *Skriptfiles.*

   ```
   deta = det(a); mineig = min(eig( a));
   ```

   sei auf einem File mit Namen `intro_1.m` gespeichert. Wird in Matlab der Befehl

   ```
   >> intro_1
   ```

   eingegeben, so werden die Anweisungen, die in `intro_1.m` enthalten sind, ausgeführt. Man erspart sich also das Eintippen der Befehle. Allerdings muss zur Durchführung im konkreten Fall die Matrix a im Arbeitsspeicher vorhanden sein.

2. *Functionfiles.* Folgendes sei im File mit Namen `intro_2.m` gespeichert:

   ```
   function [ dt, mineig ] = intro_2( a);
   dt = det(a); mineig = min(eig( a));
   ```

   Wird in Matlab der Befehl

   ```
   >> [u,v] = intro_2( X);
   ```

   eingegeben, so wird im ersten Argument auf der linken Seite, also in u der Wert der Determinante von X, und im zweiten Argument der kleinste Eigenwert von X retourniert.

Zugriff auf Teile einer Matrix. A sei eine Matrix im Arbeitsspeicher von Matlab. Die Sequenz

```
>> I = [1 2 3]; J = [4 2];
>> B = A(I,J);
```

extrahiert aus der Matrix A die Untermatrix B, bestehend aus den Zeilen 1,2,3 und Spalten 4,2 (in dieser Reihenfolge). Mit

```
>> B = A(:, 2);
```

wird die Spalte 2 von A im Vektor B abgespeichert.

Die *Lösung der Gleichung* $Ax = b$ kann im Matlab mit

```
>> x = A \ b;
```

ermittelt werden. Dies funktioniert sehr allgemein, d. h. A muss nicht quadratisch sein. Einzige Bedingung: die Anzahl der Zeilen von A muss gleich der Anzahl der Zeilen von b sein.

Falls x und y zwei Vektoren gleicher Länge sind, so liefert

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

einen *Graphen* bei dem die Punkte (x_1, y_1) mit (x_2, y_2) etc. verbunden werden.

Elementweise Operationen, z.B. $c_{ij} := a_{ij} * b_{ij}$ können in folgender Weise ausgeführt werden:

```
>> C = A .* B;
```

Der Punkt vor dem Multiplikationszeichen bedeutet elementweise Multiplikation, anstelle der normalen Matrixmultiplikation. Man könnte dies auch in einer Schleife durchführen:

```
for i=1:n; for j = 1:m; c(i,j) = a(i,j) * b(i,j); end; end;
```

dies ist aber bedeutend langsamer.

Zur *Programm-Kontrolle* gibt es die üblichen Konstrukte wie if-then-else, while, pause, break, etc.

Das *Einlesen von externen Daten* in Matlab kann auf mehrere Arten erfolgen. Hier eine einfache Möglichkeit. Angenommen, die Daten sind in einer gewöhnlichen ASCII Datei abgespeichert (eine Zahl je Zeile). Der Befehl

```
load filename
```

bewirkt, dass in Matlab ein Vektor mit dem Namen `filename` erstellt wird. Aus diesem Vektor kann man dann die einzelnen Daten extrahieren:

```
n=filename(1); m=filename(2); c=filename(3:n+2); b=filename(n+3:n+m+2);
```