Christian Clason
Blatt 0

UE 621.262
SS 2011

MATLAB – EINE KURZE EINFÜHRUNG

1 ALLGEMEINES

MATLAB (MATrix LABoratory) ist eine interaktive Umgebung für wissenschaftliche numerische Berechnung und Visualisierung. Die besondere Stärke für unsere Zwecke liegt in dem sehr einfachen *Debugging*: sämtliche Zwischenergebnisse lassen sich leicht anzeigen und manipulieren.[1]
Wichtigstes Merkmal: MATLAB kennt nur einen Datentypen: (reelle und komplexe) Matrizen. Das bedeutet:

- *Skalare* sind 1×1-Matrizen
- *Zeilenvektoren* der Dimension n sind $1 \times n$-Matrizen
- *Spaltenvektoren* der Dimension n sind $n \times 1$-Matrizen

1.1 DIE KOMMANDOZEILE

Startet man MATLAB, so sieht man zuerst eine *Kommandozeile* ('>>', OCTAVE: '`octave:1>`'). Dort kann man:

- *Befehle* eingeben: etwa '`sin(1)`', '`exp(-1)`', '`sqrt(2)`'. Das Ergebnis wird sofort angezeigt. Schließt man einen Befehl mit '`;`' ab, so wird die Ausgabe unterdrückt. Mit '`,`' kann man mehrere Befehle in einer Zeile verbinden, ohne dass die Ausgabe unterdrückt wird.
- *Variablen zuweisen*: etwa '`x = cos(pi);`', '`null = 4-4;`'. Die so zugewiesenen Variablen können in folgenden Zeilen verwendet werden: `y=x*2;`. Der Befehl '`who`' zeigt alle bekannten Variablen an. Der Wert einer Variable kann durch Eingabe des Variablennamens ausgegeben werden: '`x`', '`y`'. Der Befehl '`clear x`' löscht die Variable `x` aus dem Arbeitsspeicher.
- *Hilfe erhalten*: '`help befehl`' zeigt eine kurze Dokumentation zu `befehl`.
- *Laufzeiten messen*: '`tic; befehl; toc`' führt `befehl` aus und gibt die dafür nötige Rechenzeit aus. (Diese kann auch mit `t = toc` einer Variablen zugewiesen werden.)
- Das *Ausgabeformat* mit dem Befehl '`format`' beeinflussen (siehe `help format`). (MATLAB rechnet intern immer mit *double precision*; dies läßt sich auch nicht ändern.)
- Einen *Graphen* zeichnen: '`plot(x,y)`', wobei für die x-Achse die Komponenten des Vektors `x` und für die y-Achse die Komponenten des Vektors `y` verwendet werden (beide müssen die selbe Länge haben). '`plot(x,y1,x,y2)`' zeichnet sowohl `y1` als auch `y2` in Abhängigkeit von `x`, '`plot(y)`' ist gleichbedeutend mit '`plot(1:length(y),y)`'.
- Mit den Pfeiltasten 'auf' und 'ab' frühere Befehle zurückholen.

1.2 SKRIPTE

Möchte man eine Serie von Befehlen später wieder verwenden, so kann man sie in Form eines *Skripts* speichern. Dazu gibt man die Befehle im *Editor* ein, und speichert sie mit der Erweiterung '`.m`' ab (etwa '`meinskript.m`').

- Skripte lassen sich wie Befehle aufrufen: '`meinskript;`'. Dafür müssen sie im aktuellen Arbeitsverzeichnis liegen. Das aktuelle Verzeichnis erhält man durch Eingabe von '`pwd`', durch '`cd verzeichnis`' wechselt man in das Unterverzeichnis `verzeichnis`. Den Verzeichnisinhalt gibt '`ls`' aus.
- Skripte können mit Kommentaren versehen werden: Alles, was auf ein '`%`' folgt, wird von MATLAB ignoriert.

[1] Alle Ausführungen hier gelten—wo nicht anders angegeben—unverändert auch für OCTAVE

- Skripte haben Zugriff auf sämtliche Variablen, die auf der Kommandozeile (oder in früheren Skripten) definiert wurden. Genauso sind nach Beendigung eines Skriptes sämtliche Variablen, die im Skript definiert wurden (und Änderungen an bereits definierten Variablen) auf der Kommandozeile verfügbar.

1.3 FUNKTIONEN

Eine besondere Art von Skripten sind *Funktionen*, die neue Befehle definieren. Um ein Skript als Funktion zu deklarieren, gibt man als *erste Zeile* ein:

```
function [out1,out2,...,outn] = meinefunktion(in1,in2,...,inm)
```

Diese Funktion sollte dann als '`meinefunktion.m`' abgespeichert werden. Von der Kommandozeile (bzw. aus Skripten oder weiteren Funktionen) lässt sich die Funktion durch

```
[out1,out2,...,outn] = meinefunktion(in1,in2,...,inm);
```

aufrufen. Gibt es nur eine Ausgabevariable, so können die eckigen Klammern weggelassen werden:

```
out = meinefunktion(in);
```

Im Unterschied zu Skripten sind bei Beginn der Funktionsausführung nur die Eingabevariablen `in1,...inm` bekannt; genauso sind nach Durchführung nur noch die Ausgabevariablen verfügbar. Auch sind alle Änderungen, die die Funktion an den Eingabevariablen durchführt, nach der Ende der Funktion "vergessen".
Wird in der Funktion der Befehl '`keyboard`' eingefügt, so wird die aktuelle Funktion angehalten, sobald der Befehl erreicht wird. Man kann dann auf der Kommandozeile den aktuellen Stand sämtlicher Variablen anzeigen und verändern. Der Ablauf der Funktion wird durch den Befehl '`return`' fortgesetzt.

2 MATRIZEN

Für die *Erzeugung* von Matrizen stehen folgende Möglichkeiten zur Verfügung:

- '`A=[1,2,3;4,5,6;7,8,9]`' erzeugt
$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$
Die Eingabe erfolgt also zeilenweise: Kommata (oder Leerzeichen) trennen Spalten, Strichpunkte Zeilen.
- '`zeros(m,n)`' erzeugt eine Matrix mit `m` Zeilen und `n` Spalten, die nur 0 als Einträge hat.
- '`ones(m,n)`' erzeugt eine Matrix mit `m` Zeilen und `n` Spalten, die nur 1 als Einträge hat.
- '`rand(m,n)`' erzeugt eine Matrix mit `m` Zeilen und `n` Spalten, die als Einträge gleichverteilte Zufallszahlen zwischen 0 und 1 hat.
- '`eye(m,n)`' erzeugt eine Matrix mit `m` Zeilen und `n` Spalten, die auf der Diagonale 1 und sonst 0 als Einträge hat. (Insbesondere erzeugt '`eye(n,n)`' die `n`×`n` Einheitsmatrix.)
- '`i:j`' erzeugt den Zeilenvektor `[i,i+1,...,j-1,j]`, '`i:k:j`' den Vektor `[i,i+k,i+2k,...,i+mk]` mit $i + mk \leq j < i + (m+1)k$.

Der einfache *Zugriff* auf Matrixelemente ist die Stärke von MATLAB:

- '`A(i,j)`' ist das *Element* der Matrix `A` in der `i`-ten Zeile und `j`-ten Spalte. Für Spalten- oder Zeilenvektoren `x` liefert '`x(i)`' die `i`-te *Komponente*. (Achtung: '`A(i)`' liefert den `i`-ten Eintrag der Matrix `A`, spaltenweise gezählt (d.h. `A(2)`=4 im obigen Beispiel).)
- '`A(i1:i2,j1:j2)`' ist die *Submatrix* aus der `i1`-ten bis `i2`-ten Zeile sowie der `j1`-ten bis `j2`-ten Spalte.
- '`A(:,j)`' ist die komplette `j`-te *Spalte* von `A`; genauso ist '`A(i,:)`' ist die komplette `i`-te *Zeile* von `A`. (Mehrere Spalten bzw. Zeilen lassen sich mit '`A(:,j1:j2)`' bzw. '`A(i1:i2,:)`' auswählen.)

- ‘`size(A,1)`’ bzw. ‘`size(A,2)`’ ist die *Zeilen-* bzw. *Spaltenanzahl* von `A`; ‘`size(A)`’ liefert den Zeilenvektor [*Zeilenanzahl, Spaltenanzahl*].
- ‘`length(x)`’ gibt die Anzahl der Elemente des (Zeilen- oder Spalten-)Vektors `x` aus.
- Der Index ‘`end`’ bezeichnet den *größtmöglichen Index*: ‘`x(end)`’ ist die letzte Komponente des (Zeilen- oder Spalten-)Vektors `x`; ‘`A(:,end)`’ ist die letzte Spalte, ‘`A(end,:)`’ die letzte Zeile der Matrix `A`.

Die folgenden *Operationen* sind für Matrizen definiert:

- ‘`+,-,*`’ bezeichnet die Addition, Subtraktion bzw. Multiplikation von Matrizen (bzw. Vektoren). Dabei müssen natürlich die Dimensionen der Operanden stimmen!
- ‘`.*`’, ‘`./`’ bezeichnet die *komponentenweise* Multiplikation bzw. Division von Matrizen (bzw. Vektoren). Beispiel: `[1 2 3].*[1 2 3]` liefert `[1 4 9]`.
- ‘`^`’ ist die (Matrix-)*Potenz* (‘`A^2`’ entspricht der Matrixmultiplikation ‘`A*A`’); die komponentenweise Potenz ist ‘`.^`’ (`[1 2 3].^2` liefert also `[1 4 9]`.).
- `A'` ist die *Transponierte* der Matrix `A` (bzw. die hermitesch-konjugierte Matrix, falls `A` komplex ist). Insbesondere erzeugt `(i:j)'` einen Spaltenvektor.
- ‘`A\b`’ (der *Backslash-Operator*) gibt die Lösung `x` des linearen Gleichungssystems `Ax=b` aus.
- Die eingebauten Funktionen wie `sin` und `exp` arbeiten komponentenweise, falls ihnen eine Matrix bzw. ein Vektor übergeben wird. Beispiel: `exp([0 1 2])` liefert `[1.0000 2.7183 7.3891]`.

3 Ablaufsteuerung

Wie die meisten höheren Programmiersprachen bietet auch Matlab:

- *Bedingte Anweisungen*:

```
if (AUSDRUCK)
    ANWEISUNGEN
else
    ANWEISUNGEN2
end
```

  führt den Block `ANWEISUNGEN` aus, falls `AUSDRUCK` wahr ist, ansonsten `ANWEISUNGEN2`. Für `AUSDRUCK` wird in der Regel ein *Vergleich* stehen, etwa `x==y`, `x<y`, `x>=y` oder `x~=y` (letzteres steht für $x \neq y$). Mehrere Ausdrücke können per && (logisches ‘und’) bzw. `||` (logisches ‘oder’) verknüpft werden.

- *Schleifen*:

```
for VARIABLE = VEKTOR
    ANWEISUNGEN
end
```

  führt den Block `ANWEISUNGEN` wiederholt aus, wobei `VARIABLE` der Reihe nach als Wert die Komponenten von `VEKTOR` zugewiesen bekommt. Beispiel: `for i = 1:n`

- *Bedingte Schleifen*:

```
while (AUSDRUCK)
    ANWEISUNGEN
end
```

  führt den Block `ANWEISUNGEN` aus, solange `AUSDRUCK` wahr ist. Dabei wird `AUSDRUCK` *vor* jedem Schleifendurchlauf geprüft, insbesondere wird der Block `ANWEISUNGEN` gar nicht ausgeführt, wenn `AUSDRUCK` vor Beginn der Schleife falsch ist.

- *Vorzeitiger Abbruch:* Die Anweisung ‘`return`’ verlässt die aktuelle Schleife, bedingte Anweisung oder Funktion.