Ein kleiner Matlab Primer

Frank Schimmel

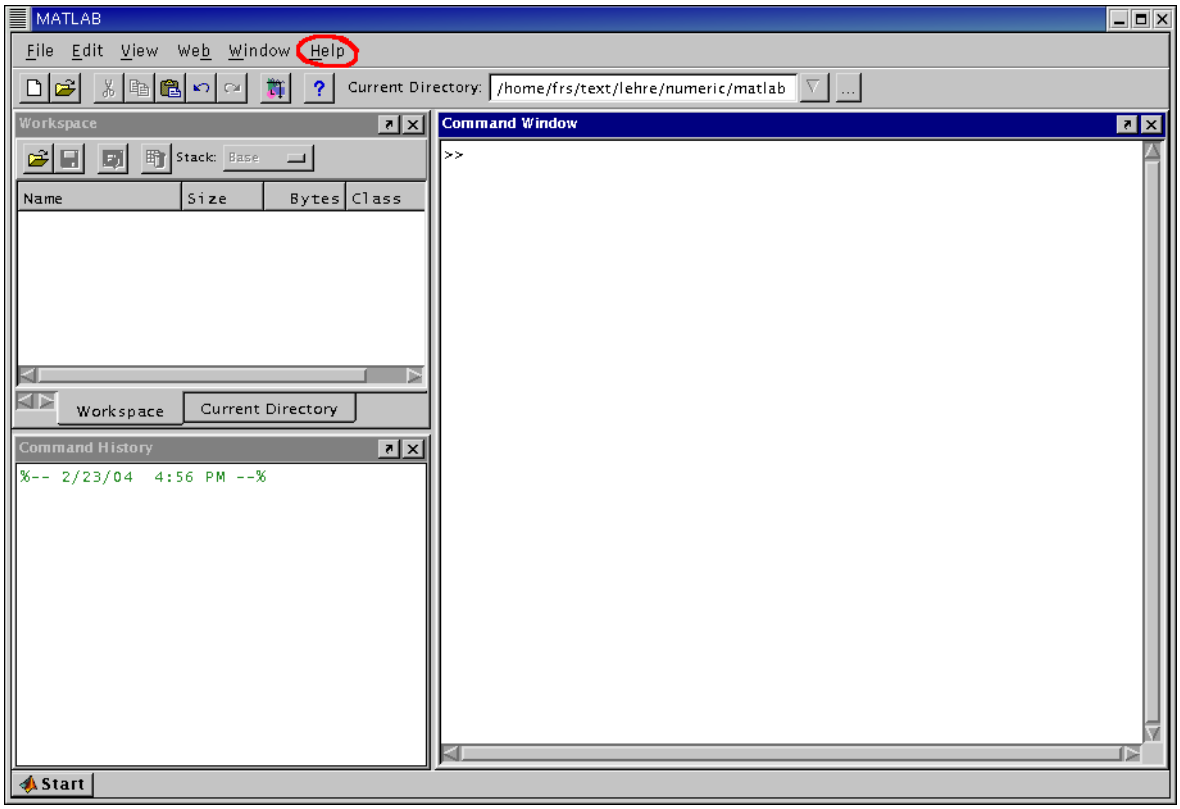
Matlab ist eine Programmiersprache für des technische und wissenschaftliche Rechnen. Mit Matlab lassen sich relativ einfach erste numerische Berechnungen realisieren und auch sofort visualisieren. Dieses Dokument versucht einen erleichterten Einstieg in Matlab zu gewärleisten.

Der Name Matlab steht übrigends für ***Mat****rix* ***Lab****oratory*.

Inhaltsverzeichnis

1 Der Umgang mit Matlab

Nach dem Starten von Matlab ergibt sich etwa folgendes Bild:



Das Hauptfenster teilt sich in drei Unter-Fenster, *Workspace*, *Command History* und *Command Window*. In letzteres können Kommandos (Ausdrücke) eingegeben werden, die Matlab dann auswertet. Die eingegebenen Ausdrücke werden im Fenster *Command History* protokolliert (mitgeschrieben), und stehen somit später erneut zur Verfügung. Das Kommando-Protokoll (oder Ausschnitte davon) können auch gespeichert werden.

Das *Workspace*-Fenster gibt einen Überblick über alle erzeugten Variablen (keine zu Begin). Variablen werden automatisch durch eine Zuweisung auf diese auf diese erzeugt, so z.B. durch die Eingabe von `A = [[1 2 3 4]; [5 6 7 8]]`. Diese quittiert Matlab zum einen mit einer Ausgabe des Ergebnissen im `Command Window` und zum anderen mit einem Eintrag im *Workspace* für die Variable `A` mit der Größe 2×4 (eine Matrix). Zu beachten ist die Verwendung von Leerstellen, um mehrere Spalten, und des `;`, um mehrere Zeilen zu erhalten. Wird ein Ausdruck keiner neuen oder alten Variable zugewiesen, so speichert Matlab dieses Ergebnis in einer automatisch angelegten Variable `ans` (kurz für `answer`). So Ergibt die Eingabe von `42` im *Command Window* den Eintrag `ans` mit der Größe 1×1 (Skalare Zahl) im *Workspace*.

1.1 Hilfe

Matlab hat auch ein eingebautes `help`-Kommando. Dieses gibt die wesentlichen Informationen über ein bestimmtes Thema (z.B. `matlab/general` über algemeine Kommandos oder `matlab/elmat` über Matrixmanipulationen), eine Funktion oder eingebaute Variable im *Command Window* aus. Sofern selbst erstellte Funktionen (Abschnitt 3) entsprechend mit Kommentaren versehen wurden, funktioniert dies ebenso für solche. Es sei auch auf die äußerst ausführliche Online-Hilfe erwähnt, die über das entsprechende Menü (siehe Abbildung) oder das Kommando **helpbrowser** erreichbar ist.

1.2 Matlab verlassen

Matlab wird durch die Eingabe von `quit` oder `exit` im *Command Window* oder über den entsprechenden Menüpunkt wieder beendet.

2 Syntax von Kommandos und Ausdrücken

Die Syntax von Matlab ist zeilenorientiert: ein Zeilenumbruch beendet normalerweise ein Kommando. Um ein Kommando in der nächsten Zeile fortzusetzen, muß diese mit drei Punkten (`...`) enden. Dies ist vor allem für die Lesbarkeit von Skripen und Funktionen wichtig (Abschnitt 3). Um mehrere Kommandos in einer Zeile unterzubringen, können diese durch ein `,` getrennt werden. Die Ausgabe des Ergebnisses eines Ausdrucks kann durch das Anfügen eines `;` am Ende des Ausdruckes unterdrückt werden.

2.1 Kommentare

Es konnen – und sollten – auch Kommentare in Funktionen eingefügt werden. Kommentare beginnen mit einem Prozentzeichen (`%`) und erstrecken sich bis zum Ende der Zeile. Ein Beispiel ist in Abschnitt 3 gegeben.

2.2 Kontrollstrukturen

Matlab bietet die üblichen Kontrollstrukturen, deren Syntax im Folgenden kurz zusammengefaßt wird. Dabei stellt *`geneigter Text`* einen Platzhalter dar.

Zum einen gibt es einfache Fallunterscheidungen:

```
if logischer Ausdruck
  Kommandos
elseif logischer Ausdruck
  Kommandos
```

```
else
  Kommandos
end
```

wobei die `elseif`- und `else`-Teile optional sind. Weiterhin gibt es ein Konstrukt für mehrfache Fallunterscheidungen:

```
switch Ganzzahl- oder Zeichenkettenausdruck
case Wert1
  Kommandos
case Wert2
  Kommandos
.
.
.
otherwise
  Kommandos
end
```

Es werden zwei Konstrukte für Iterationen (Schleifen) geboten: `while`-Schleifen, die ausgeführt werden solange eine beliebige Bedingung erfüllt ist,

```
while logischer Ausdruck
  Kommandos
end
```

und `for`-Schleifen, in denen eine ganzzahlige Schleifenvariable hoch- oder runtergezählt wird:

```
for Indexname = Startwert : Schrittweite : Endwert
  Kommandos
end
```

2.3 Matrizen und Vektoren erzeugen

Matrizen und Vektoren können auf verschiedene Arten erzeugt werden. Dabei sind Vektoren für Matlab nichts anderes als Matrizen, die nur eine Zeile bzw. Spalte haben.

Matratzen können direkt mit eckigen Klammern ausgedrückt werden. Beispiel: `A = [[1 2 3 4]; [5 6 7 8]]`.

Eine einfache Möglichkeit einen Vektor zu erzeugen ist der Doppelpunktoperator (siehe auch 2.4): `x = [0:1/20:5]` erzeugt einen Vektor der Länge 101, dessen benachbarte Elemente alle die Differenz 1/20 haben.

Weiterhin nützlich sind die Funktionen `ones` und `zeros`. Diesen wird die größe der zu erzeugenden Matrix übergeben, die dann mit 1 bzw. 0 belegt wird.

2.4 Ausdrücke

Zum Formen von Ausdrücken bietet Matlab neben den üblichen **Verknüpfungen**, `*`, `/`, `+`, `-` und `^` (Exponentiation, `a^b` bedeutet a^b), gibt es noch die logischen Operatoren `&` (logisches und) und `|` (oder) sowie die Relationen `==` (Gleichheit),[1] `~=` (Ungleichheit), `<`, `>`, `<=` und `>=`. Weiterhin ergibt `A'` die adjungierte (transponierte, konjugiert komplexe) Matrix einer Matrix `A` und `A'` die Transponierte.

Den multiplikativen Operatoren, `*`, `/` und `^`, kann auch ein Punkt (`.`) vorangestellt werden. Dann wird statt der normalen Matrix-Multiplikation um eine elementweise Verknüpfung der Matrizen vorgenommen: `(1:5) * (1:5)'` ergibt den skalaren Wert 55, `(1:5)' * (1:5)` ergibt eine 5×5-Matrix und `(1:5) .* (1:5)` ergibt den Vektor `[1 4 9 16 25]`.

Ein vollständige Liste der eingebauten Operationen gibt das Kommando `help matlab/ops`. Die Kommandos `help arith` und `help relop` geben ausführlichere Erklärungen der arithmetischen Operationen bzw. Relationen. Siehe auch Tabelle 1

Funktionen werden durch Anfügen einer Liste von Argumenten (in Klammern und durch Kommata getrennt) auf diese angewandt, z.B. ergibt `exp(1.0)` die Eulersche Zahl und `rand(3, 7)` eine 3×7 Matrix, deren Einträge Zufallszahlen sind. Fast alle Funktionen in Matlab lassen sich auch (elementweise) auf Matrizen anwenden.

[1] Achtung! Es besteht ein wichtiger Unterschied zwischen einer Zuweisung eines Wertes zu einer Vatiablen (`A = B`) und dem Vergleich zweier Variablen (`A == B`).

Arithmetische Operatoren	
`+`,`-`	Addition, Subtraktion (elementweise)
`*`	Matrix-Multiplikation
`.*`	elementweise (Array-) Multiplikation
`/`, `\`	„Matrix-Division“
`./`	elementweise (Array-) Division
`^`	Matrix-Potenz
`.^`	elementweise (Array-) Potenz
Logische Operatoren	
`<`, `<=`	kleiner(gleich) als
`>`, `>=`	größer(gleich) als
`==`, `~=`	gleich, ungleich
Sonstige Operatoren	
`'`	Adjungierte (konjungiert-komplexe Transposition)
`.'`	Transponierte
`:`	Bereiche: `a:b` ergibt die Sequenz der Zahlen von `a` bis `b`, `a:c:b` ergibt die Sequenz der Zahlen von `a` bis `b` mit dem Inkrement `c`

Tabelle 1: Operatoren in Matlab

Auf **einzelne Elemente einer Matrix** kann mit einer änlichen Notation wie dem Funktionsaufruf zugegriffen werden: durch Indexierung mit runden Klammern. Ist `A` eine Matrix, dann ergibt `A(2, 5)` das Element von `A` in der zweiten Zeile und fünften Spalte. **Untermatrizen** können mit dem Doppelpunktoperator erzeugt werden: `A(2:4, 3:7)` ergibt eine 3×5-Matrix aus der zweiten bis vierten Zeile und dritten bis siebenten Zeile von `A`. Alle Zeilen oder Spalten können durch einen blanken Doppelpunkt (`:`) gewonnen werden. Bei Vektoren braucht nur ein Index angegeben werden. Werden einem oder mehreren Elementen, die außerhalb des schon definierten Indexbereiches einer Matrix liegen, neue Werte zugewiesen, so wird die Matrix **automatisch erweitert**. So ergibt `A = ones(2, 3)` eine 2×3-Matrix von Einsen und ein darauffolgendes `A(3,3:4) = 42` ergibt

```
A =
   1   1   1   0
   1   1   1   0
   0   0  42  42
```

3 Eigene Funktionen erstellen und verwenden

Neben dem großen Angebot an mitgelieferten Funktionen, können auch leicht eigene Funktionen oder Skripte erstellt und benutzt werden. Dies sind einfache Dateien, die eine Abfolge von Kommandos und Ausdrücken beinhalten. Die Namen dieser Dateien sollten mit `.m` enden (*M files* in der Matlab-Sprechweise). Auch sollten solche Programm-Dateien der Einfachheit halber immer im aktuellen Verzeichnis abgelegt werden. Im Falle eines einfachen Skriptes werden die Kommandos nacheinander abgearbeitet, als wären sie im *Command Window* eingegeben worden. Funktionen könne auch Parameter mitgegeben werden und sie liefern ein oder mehrere Ergebnisse zurück.

Um eine Funktion zu definieren, muß die Datei als erstes eine entspechende `function`-Deklaration enthalten. Eine solche Datei sieht etwa so aus:

```
function result = foo(parameter_1, parameter_2, parameter3)
% FOO - Diese Funktion ist nur ein Beispiel.
%   Eine genauere Beschreibung der Funktion, ihrer Paremeter und
%   des Ergebnisses sollte hier erfolgen...

% Zweisen des Funktionsergebnisses
% auf die Ergebnisvariable 'result':
result = parameter_1 + parameter_2 + ...
         parameter3 + 42;
```

Die Datei muß `foo.m` heißen[2] und die Funktion kann mit z.B. `foo(7, 5, 13)` (ergibt 67) verwendet werden.

[2] Andernfalls findet Matlab die Funktion nicht.

Eine Funktion kann auch mehrere Ergebnisse erzeugen:

```
function [result1, result2] = bar(parameter_1, parameter_2)
% BAR - Diese Funktion ist ein weiteres Beispiel.
%   Eine genauere Beschreibung der Funktion, ihrer Paremeter und
%   Ergebnisse sollte hier erfolgen...

% Diese Funktion hat zwei Ergebnisse:
result1 = parameter_1 + 42;
result2 = parameter_2 + 42;
```

Diese können dann z.B. im *Command Window* in der Form `[a, b] = bar(7, 19)` verwendet werden. Dies weist den Variablen `a` und `b` die Werte 49 bzw. 61 zu.

Ein *M file* kann auch mehrere Funktionen enthalten. Davon ist aber nur eine von außen ‚sichtbar': diejenige, die genauso heißt wie die Datei (ohne `.m` natürlich). Alle anderen können nur innerhalb dieser Haupfunktion benutzt werden.

Und vergessen Sie nicht, ihre Funktionen zu kommentieren (Abschnitt 2.1)!

4 Alternativen zu Matlab

Eine Alternative zu Matlab stellt das GNU Octave (`http://www.octave.org/`) dar. Octave ist weitgehend kompatibel zu Matlab. Jedenfalls habe ich keine Probleme mit meinen Programmen erfahren. (Nur Plots sehen ein wenig anders aus.) Octave ist freie Software.[3] Fertige Binaries gibt es für Linux (Debian, SuSE, und Redhat bieten jeweils eigene Pakete), für Windows (unter `http://sourceforge.net/projects/octave/`, dann *octave-forge-windows*) und Mac OS X (als Teil des fink-Projekts, `http://fink.sourceforge.net/`). Alles ohne Gewähr; ich habe selbst kompiliert...

[3] http://www.gnu.org/philosophy/free-sw.html