Dr. Michael Lehn
Iris Häcker
Klaus Stolle
Institut für Numerische Mathematik
Universität Ulm

WS 2013/2014

Einführung in Matlab

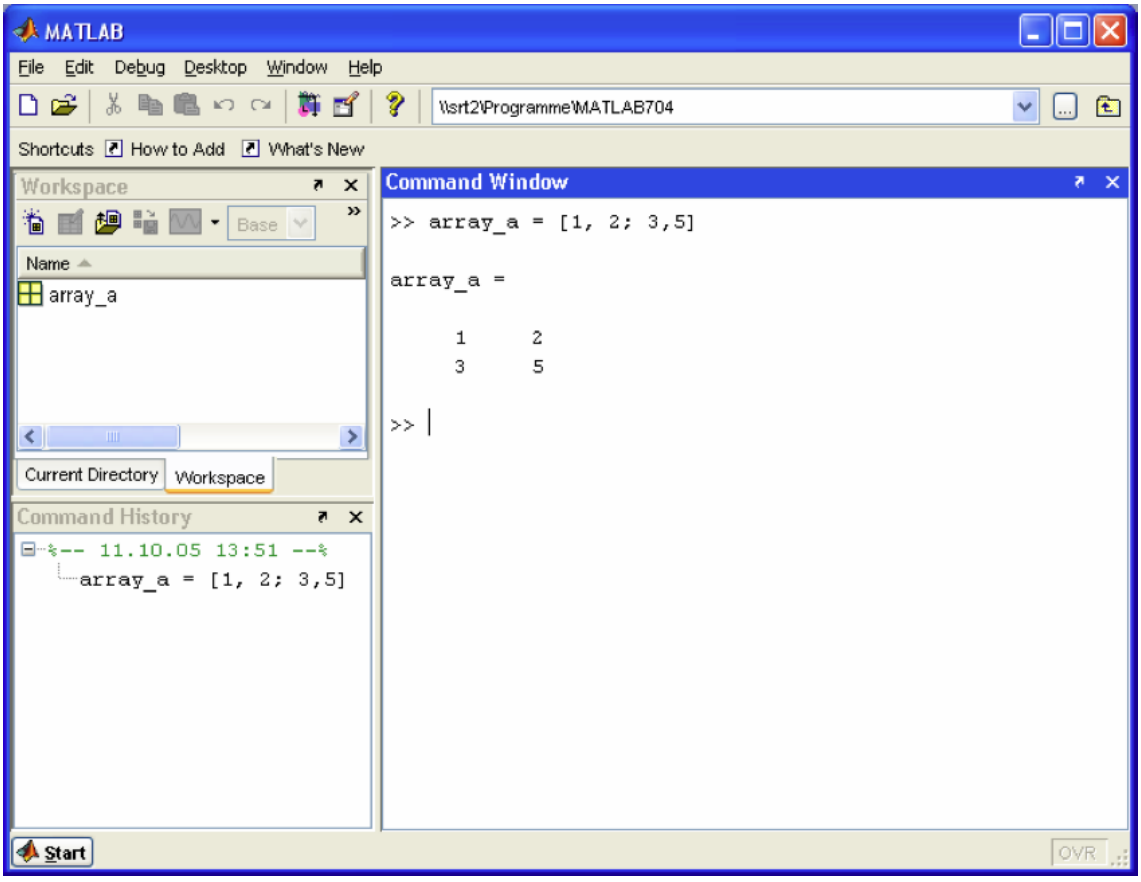
1 Grundlegendes

Matlab = MATrix LABoratory.
Programmiersprache für wissenschaftlich-technisches Rechnen.
Matlab wurde Ende der siebziger Jahre von Cleve Moler entwickelt, aufbauend auf Unterprogrammen von LINPACK und EISPACK (Software-Pakete für Lineare Algebra, in Fortran geschrieben). Die heutige Matlab-Software basiert auf der Programmiersprache C und wird von der Firma *The MathWorks* hergestellt und vertrieben.

2 Benutzerführung von Matlab

2.1 Die Benutzeroberfläche

Nach dem Start von Matlab wird - Standardeinstellungen vorausgesetzt - eine dreigeteilte Benutzeroberfläche angezeigt:



• In dem großen, rechts stehenden Fenster (Command Window) werden die eigentlichen Berechnungen ausgeführt. Hier können die Matlab-Befehle über eine Kommandozeile eingegeben und mit <return> ausgewertet werden.
• Links oben steht die Workspace-Anzeige, in der alle aktuell gespeicherten Variablen dargestellt sind.
• In der Command-History (links unten) werden alle bisher verwendetet Befehle festgehalten

2.2 Wichtige Einstellungen

Zwei Voreinstellungen sind für die Arbeit mit Matlab wichtig:

1) In der Eingabezeile rechts oben in der Benutzeroberfläche ist der Pfad des aktuell gewünschten Arbeitsverzeichnisses einzutragen (bzw. auszuwählen).

2) Über den Menüpunkt *File* → *Set Path* kann ein Fenster geöffnet werden, in dem alle zusätzlichen Pfade einzutragen sind, auf die ein Zugriff möglich sein soll. Einstellungen müssen hier nur bei Bedarf vorgenommen werden.

2.3 Schnelle Hilfe

Über die Eingabe "`help` Befehlsname" erhält man direkt im Command Window Informationen zu einem Matlab-Befehl. Außerdem gibt es natürlich auch die übliche Hilfefunktion über das Auswahlmenü. Hier findet man häufig ausführlichere Informationen.

3 Grundelemente und -funktionen von Matlab

3.1 Variablen

Variablen müssen normalerweise nicht gesondert deklariert werden. Sie werden durch eine Wertzuweisung erzeugt und sind automatisch vom Typ des ihnen zugewiesenen Wertes. Z.B.:

```
>> a = 5
a =
    5
```

oder:

```
>> b = 'teststring'
b =
teststring
```

Selbstverständlich kann der Wert einer Variablen einer anderen Variablen übergeben werden:

```
>> a = b
a =
teststring
```

Beachte: Nach jeder Befehlseingabe wird direkt unter der Befehlszeile das Ergebnis der Befehlsauswertung angezeigt. Durch ein Semikolon am Ende des Befehls kann diese Anzeige unterdrückt werden.

```
>> a = 5;
```

3.2 Ausführen von Rechenoperationen

Rechenoperationen können auf Variablen oder auch auf Zahlenwerte angewendet werden:

```
>> c = a*5
c =
    25
>> a+c
ans =
    30
```

Beachte: Wird das Ergebnis einer Befehlseingabe vom Benutzer nicht explizit einer Variablen zugeordnet, so wird es von Matlab standardmäßig der Variablen `ans` (**ans**wer) zugewiesen. Auf `ans` kann der Benutzer wie auf jede andere Variable zugreifen:

```
>> 17
ans =
    17
>> 2*ans
ans =
    34
```

3.3 Rechnen mit Vektoren und Matrizen

Eine der großen Stärken von Matlab ist seine matrizenorientierte Arbeitsweise.
Matrizen (und damit auch Vektoren) können, genau wie skalare Zahlenwerte, als Variablen gespeichert werden:

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

>> B = [1, 3; 4, 2; 1, 7]
B =
     1    3
     4    2
     1    7

>> y = [7 8 9]
y =
     7    8    9
```

Bei der Eingabe werden die Elemente einer Zeile durch Kommas oder Leerzeichen voneinander getrennt. Durch Semikolon werden die Zeilen voneinander getrennt.
Es gibt darüber hinaus viele verschiedene Möglichkeiten, Matrizen aufzustellen. Z.B.:

```
>> zeros(2,3);  % Matrix mit Null-Eintraegen; mit 2 Zeilen und 3 Spalten
>> ones(5,4);   % Matrix mit Eins-Eintraegen; mit 2 Zeilen und 3 Spalten
>> eye(5);      % Einheitsmatrix mit 5 Zeilen (und Spalten)
>> 1:5;         % Zeilenvektor mit den Eintraegen 1, 2, 3, 4, 5
>> 1:0.5:5;     % Zeilenvektor mit den Eintraegen 1, 1.5, 2, 2.5, ... , 4.5, 5
```

Beachte: Kommt in der Befehlszeile ein Prozentzeichen (%) vor, so wird der dahinter stehende Text als Kommentar interpretiert und dementsprechend ignoriert.
Auf die einzelnen Elemente der Matrizen kann folgendermaßen zugegriffen werden:

```
>> A(3,1)
ans =
     6
```

→ Element der 3. Zeile und der 1. Spalte von A.
Man kann auch komplette Zeilen oder Spalten als Vektoren herausgreifen. Hier zum Beispiel die zweite Zeile von B:

```
>> B(2,:)
ans =
     4    2
```

Die "Größe" einer Matrix (also die Zeilen und Spaltenzahl) erhält man mit dem Befehl `size()`:

```
>> size(A)
ans =
     3    3
```

Die "Länge" eines Vektors kann entsprechend über den Befehl `length()` ermittelt werden:

```
>> length(y)
ans =
     3
```

Mit Matrizen können Rechenoperationen in derselben Weise ausgeführt werden, wie mit skalaren Größen:

```
>> A*B
ans =
     15    41
     15    23
     24    40
```

Es müssen dabei lediglich die üblichen Regeln der Matrizenrechnung berücksichtigt werden, z.B. bezüglich der Zeilen- und Spaltenzahl bei der Multiplikation.
Die Transposition einer Matrix (eines Vektors) wird durch ein angehängtes Apostroph erreicht:

```
>> y_column = y'
y_column =
     7
     8
     9
```

Will man die Rechenoperationen nicht im Sinne der Matrizenrechnung, sondern elementweise ausführen, so muss vor das Operatorzeichen in der Regel ein Punkt eingefügt werden:

```
>> A .* [1 2 5; 3 3 4; 0 1 0]
ans =
     3    4    20
     3    9     8
     0    4     0
```

Beachte: Eine Übersicht über die verfügbaren Operationen erhält man z.B. mit dem Befehl "`help *`".

Eine typische Anwendung der Matrizenrechnung ist das Lösen von linearen Gleichungssystemen der Form $\mathbf{Ax} = \mathbf{y}$. Die Matrix $\mathbf{A}$ sei hierbei regulär und quadratisch und $\mathbf{y}$ sei ein bekannter Spaltenvektor. Der unbekannte Spaltenvektor $\mathbf{x}$ kann dann einfach bestimmt werden mittels des Befehls

```
>> x = A\y'
x =
    -0.2381
     2.1905
     0.8333
```

Der weiter oben definierte Vektor y muss hier transponiert werden, um einen Spaltenvektor zu erhalten.

Die komfortable Handhabung von Matrizen und Vektoren macht Matlab zu einem idealen Tool für die Numerik-Module. Die Eigenwerte der Matrix $\mathbf{A}$ können beispielsweise mit dem Befehl

```
>> eig(A)
ans =
     8.9291
    -2.6826
     1.7534
```

bestimmt werden.

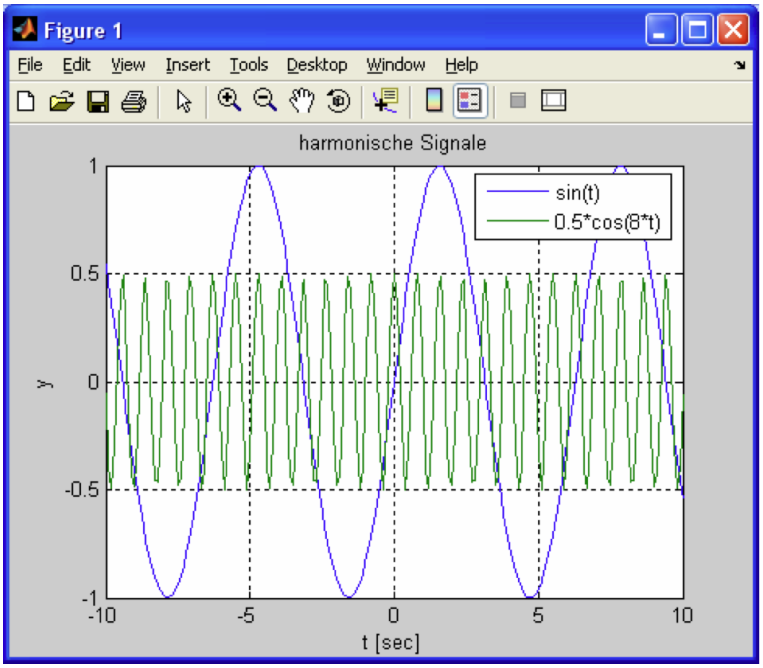
3.4 Visualisierung

Eine weitere Stärke von Matlab ist die grafische Veranschaulichung von Daten. Es steht hierzu eine Vielzahl von verschiedenen Darstellungsmöglichkeiten zur Verfügung.

Möchte man z.B. zeitabhängige Messreihen im zweidimensionalen Koordinatensystem darstellen, so wird dies mit dem Befehl `plot()` ausgeführt. Im Folgenden wird eine mögliche Anwendung dieses Kommandos mit vollständiger Beschriftung des erstellten Koordinatensystems gezeigt:

```
>> t = -10:0.1:10;    % t sei ein Vektor mit den Eintraegen -10, -9.9, -9.8, ... 9.9, 10
>> x = sin(t);        % x ist ein Vektor, da der sin()-Befehl elementeweise ausgefuehrt wird
>> y = 0.5*cos(8*t);
>> plot(t, x, t, y);
>> grid on
>> title('harmonische Signale');      % Titel ueber das Koordinatensystem schreiben
>> legend('sin(t)', '0.5*cos(8*t)');  % Kurven beschriften
>> xlabel('t [sec]');                 % x-Achse beschriften
>> ylabel('y');                       % y-Achse beschriften
```

Ergebnis:



Der Plot wird automatisch in das Grafikfenster mit der Nummer 1 (Figure 1) gezeichnet. Will man weitere Plots gleichzeitig in verschiedenen Fenstern anzeigen, so muss man für jeden zusätzlichen Plot explizit ein neues Grafikfenster öffnen. Z.B.:

```
>> figure(2);
>> plot(t, x.*y);
```

Für logarithmische Skalierungen der Achsen stehen die Plot-Befehle `semilogx()`, `semilogy()` und `loglog()` zur Verfügung.
In ähnlicher Weise ist auch die Darstellung von 3D-Plots mit den Befehlen `surface()` und `plot3()` möglich.

4 Erstellen größerer Programmstrukturen

4.1 Funktionen

Für größere Rechenaufgaben und häufig wiederkehrende Befehlsfolgen können mehrere Matlab-Befehle in einer Textdatei, der sogenannten **Funktion**, mit der Dateiendung `.m` zusammengefasst werden.
Bei Funktionen wird am Anfang der Datei eine Definition der Funktion in der Form

```
function rueckgabevariable = funktionsname(var1, var2, ...)
```

erwartet. Vorteil der Funktionen ist, dass Sie die Möglichkeit einer strukturierten Programmierung bieten. Die innerhalb der Funktion verwendeten Variablen sind nur lokal gültig (sofern nicht anders deklariert). Außerdem können Variablen aus der übergeordneten Programmstruktur nicht verändert werden.
Eine neue Funktionsdatei kann über den Menüpunkt *file* → *new* → *M-file* von der Matlab-Benutzeroberfläche aus geöffnet werden.
Beachte: Matlab arbeitet als Interpreter und nicht etwa als Compiler, wie es z.B. bei den Programmiersprachen C oder Java der Fall ist. Zu Matlab-Funktionen werden also keine Maschinencode-Dateien erstellt. Die Befehlsfolgen werden vielmehr bei jedem Aufruf neu "interpretiert".

4.2 Befehle zur Flusskontrolle

In Matlab können, ähnlich wie in den meisten anderen Programmiersprachen, Strukturbefehle wie `if`, `case` oder `for` verwendet werden (Einzelheiten zur Syntax: Siehe Matlab-Hilfe). Eingesetzt in Matlab-Routinen erlauben diese Kommandos den Aufbau von komplexen Programmenstrukturen.

4.3 Beispielroutine

Hier ein Beispiel für eine selbst geschriebene Matlab-Funktion:

```
function [arithm_mittel geom_mittel] = mittelwerte(x_vec)
% Berechnet fuer einen beliebigen Vektor x_vec das arithmetische
% und das geometrische Mittel seiner Eintraege. Die Rueckgabe der Ergebnisse
% erfolgt in einem Vektor.

arithm_mittel = sum(x_vec)/length(x_vec);
geom_mittel = prod(x_vec)^(1/length(x_vec));
```

Der Aufruf dieser Funktion erfolgt in der Art

```
>> [m_arith m_geom] = mittelwerte([1 2 3 4 5])
m_arith =
    3
m_geom =
    2.6052
```

5 Weitere Informationsquellen

• über das Help-Menü in der Matlab-Benutzeroberfläche

• auf der Hersteller-Homepage *www.mathworks.de*

• auf Seiten im Internet, die man z.B mit der Suchaufforderung "Matlab" und "Einführung" oder "Tutorial" finden kann

Beispiele zur Anwendung von Matlab erhält man über den Menüpunkt *Help* → *Demos* in der Matlab-Benutzeroberfläche.