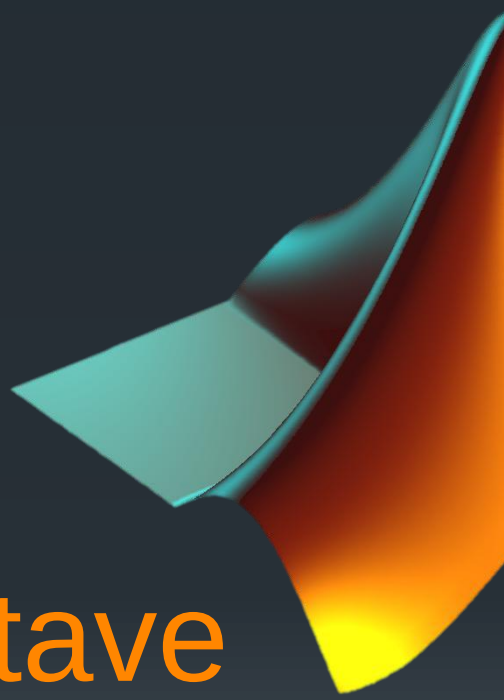


# MATLAB® / Octave

Getting Started

# Organisatorisches und Allgemeines

# Organisatorisches

- Dominik Haas (haas_dominik@yahoo.de)
- Jan Seyler (Jan_Seyler@chefmail.de)

- Mo & Do jeweils 14:00 – 17:00 Uhr
- OMZ PC-Pools

# Logins

- Benutzername: num0wXXX (ab num0w027)
- Startpasswort: NumMath00

- Passwort ändern:

```
Terminal öffnen ->
passwd <ENTER>
```

# Organisatorisches

- Zulassungskriterium: 50% erfolgreich bearbeitet
- Abgabe in Gruppen (2-3 Personen)
- Anwesenheitspflicht, falls Zusatz „hat an Programmierübungen teilgenommen“ erwünscht

- Taucht zweimal die gleiche Lösung auf, bekommt niemand Punkte
  - Punkte können auch im Nachhinein aberkannt werden.

# Umfrage ☺

- Wer hat Programmiererfahrung?
- Wer hat Zugang zu MATLAB®?
- Windows, MacOS oder Linux?

# Grundlegendes zu MATLAB und Octave

# Was ist MATLAB® ?

- „MATrix LABoratory“
- Cleve Moler

- High-Performance Programmiersprache für wissenschaftliches Rechnen (v.a. unter Ingenieuren)
- Benutzerfreundliche Oberfläche
- Vordefinierte Datentypen (z.B. für Matrizen, Vektoren,…)
- Visualisierungsmöglichkeiten
- Keine aktive Speicherverwaltung nötig

# Alternative: GNU Octave

- MATLAB ® Clone (mit Einschränkungen)
- Open Source
- Aktive Entwicklergemeinde
- Kommandozeilenbasiert
  - Grafische Oberflächen vorhanden
    - XOctave
    - QtOctave
- Numerische Bibliothek für C++

# Woher bekomme ich GNU Octave?

- Octave:
- http://www.gnu.org/software/octave
- Octave für Windows:
- http://octave.sourceforge.net

- XOctave:
- http://xoctave.webs.com/
- QtOctave:
- http://qtoctave.wordpress.com/download/

# Wie starte ich das Programm?

- Unter Windows werden Icons erzeugt…
- Unter Linux durchsucht man entweder die Programmsammlung oder öffnet das Terminal und gibt dort ‚matlab‘ bzw. ‚qtoctave‘ ein.

- Beenden: Matlab mit ‚quit‘, Octave mit ‚exit‘

# Wie bekomme ich Hilfe?

- Ausführliche online Dokumentationen
- Hilfe in MATLAB ® besser
  - help <function>
  - doc <function>
  - helpdesk
- In Octave
  - help <function>
  - doc <function>

- http://www.mathworks.com/help/documentation-center.html
- http://www.mathworks.com/help/techdoc/

# Eingabe von Zahlen, Matrizen und Vektoren

- Man kann direkt rechnen z.B.

```
((1003-29)/47)^(-5)
ans = 2.6163e-007
```

  - Die Ausgabe wird in der Variablen `ans` gespeichert.
- Der wichtigste Datentyp sind Matrizen. Im folgenden Beispiel ordnen wir der Variablen `A` eine Matrix zu:

```
A=[16 3 2 13; 5 10 11 8; 9 6 7 12; 4 15 14 1]
A=
        16    3    2   13
         5   10   11    8
         9    6    7   12
         4   15   14    1
```

# Arbeiten mit Funktionen

```
cos(pi)
ans = -1

cos(0.5)
ans = 0.87758

sinh(2*pi/3)
ans = 3.9987
```

# Matrizen definieren und zugreifen

- Zahlen werden in einer Zeile mit Blanks oder Kommata getrennt, die einzelnen Zeilen mit Semikolon
- Auf einzelne Elemente $A_{i,j}$ der Matrix werden über `A(i,j)` adressiert.
  - Es geht auch `A((j-1)*n+i)` (Matrix wird als langer Spaltenvektor interpretiert)

# Rumprobieren

- Am besten lernt man den Umgang durch einfaches Rumprobieren. Bitte berechnet: (Tipp: helpdesk)
  - Die Summe der letzten Zeile

    `sum(A(4,1:4))` oder `sum(A(end,:))`
  - Die Summe jeder Spalte

    `sum(A)`
  - Die Summe der Diagonalen von A

    `sum(diag(A))`
  - Die Summe der Antidiagonalen

    `sum(diag(fliplr(A)))`
  - Die Transponierte der Matrix A

    `A'`

# Der Doppelpunkt-Operator

- Einer der wichtigsten Operatoren in MATLAB ®
- Erscheint in verschiedenen Formen:

```
1:10
ans = 1   2   3   4   5   6   7   8   9   10

100:-7:50
ans = 100   93   86   79   72   65   58   51

0:pi/4:pi
ans = 0   0.7854   1.5708   2.3562   3.1416
```

# Ergebnisse?

- Es handelt sich um ein magisches Quadrat.
- Einfacher zu erzeugen über:

```
B = magic(4)
```

- Entspricht nicht ganz der Ausgangsmatrix. Dies korrigiert man über ein Umordnen der Spalten via:

```
C = B(:,[1,3,2,4])
```

# Etwas systematischer…

- Als Matrix Element kann jeder gültige Ausdruck verwendet werden:

```
x=[-1.3 sqrt(3) (1+2+3)*4/5]
x =
     -1.3000 1.7321 4.8000
```

- Auf Matrixelemente wird mit der Angabe des Indizes in Klammern zugegriffen (beginnend bei 1!!)

```
x(4)=42        % gab es vorher noch nicht
x =
     -1.3000 1.7321 4.80000 42.0000
```

# Weitere Grundlagen

- Komplexe Zahlen werden mit einem i oder j als imaginäre Einheit eingegeben

```
A=[1 2; 3 4]+i*[5 6; 7 8]
A =
      1.0000 + 5.0000i 2.0000 + 6.0000i
      1.0000 + 7.0000i 4.0000 + 8.0000i
```

- Beispiel:

```
exp(i*(1+i))
ans = 0.1988 + 0.3096i
```

# Matrixfunktionen

- Grundsätzlich sind alle Rechenoperationen für Matrizen definiert:
- + - * sind die bekannten Matrixoperationen
- Es ist sogar erlaubt Matrizen zu teilen. Dafür gibt es zwei Operatoren
  - `X=B/A` entspricht der Lösung der Gleichung $X * A = B$
  - `X=B\A` entspricht der Lösung der Gleichung $A * X = B$
- Der Potenzoperator `^` entspricht dem Analogon aus einer Dimension
- Für komponentenweise Rechnung muss man der Operation einen Punkt (.) voranstellen

# Übersicht über workspace

- Die momentan definierten Variablen können jederzeit mit Hilfe des Kommandos ‚`who`‘ aufgerufen werden
- Das Kommando ‚`whos`‘ ergibt eine detailliertere Übersichtstabelle
- Um all definierten Variablen zu löschen gibt man `clear all` ein
- um z.B. nur die Variable `x` zu löschen wird `clear x` eingegeben

# Programmiergrundlagen

- MATLAB ® bietet mathematische Ausdrücke, die zusammengesetzt sind aus
  - Variablen
  - Numerischen Konstanten
  - Operatoren
  - Funktionen

# Variablen

- Keine Typdeklaration
- Keine Dimensionierung
- Keine Speicherallokation
- Variablenamen bestehen aus einem Buchstaben gefolgt von beliebigen Zahlen, Buchstaben, und Unterstrichen (_)
  - Maximale Länge sind 31 Zeichen
  - MATLAB ® ist case-sensitive

# Numerische Konstanten

- Herkömmlich Notation wird verwendet
  - `3`
  - `3.14159`
  - `-99`
  - `6.02252e23`
  - `3e5i`
- Genauigkeit von ~16 Nachkommastellen
- Spanne: $10^{-308} - 10^{308}$

# Operatoren

- Wir haben bereits alle Operatoren kennengelernt, hier nur eine kurze Übersicht:

| Symbol | Bedeutung |
|---|---|
| + | Addition |
| - | Subtraktion |
| * | Multiplikation |
| / | Division |
| \ | Links-Division |
| ^ | Potenz |
| ‘ | Komplexe Konjugation, Transponieren |
| () | Festlegen der Reihenfolge |

# Funktionen

- Die meisten mathematischen Funktionen werden bereits unterstützt. Wichtige Beispiele sind:

| Funktion | Bedeutung |
|---|---|
| `abs()` | Betrag |
| `sqrt()` | Wurzel |
| `exp()` | Exponentialfunktion |
| `sin()` | Sinus |

- MATLAB ® liefert eine Auflistung aller elementaren Funktionen über den Befehl: `help elfun` bzw. `(help specfun; help elmat)`

# Konstanten

- Liste einiger nützlicher Konstanten in MATLAB ®

| Konstante | Wert |
|---|---|
| `pi` | 3.14159265… |
| `i` | Imaginäre Einheit |
| `j` | Äq. zu i |
| `eps` | Relative floating-point Genauigkeit, $2^{-52}$ |
| `realmin` | Kleinste float Zahl, $2^{-1022}$ |
| `realmax` | Größte float Zahl, $(2-\varepsilon)2^{1023}$ |
| `inf` | Unendlich |
| `nan` | Not-a-number |

# Ausgabe von Ergebnissen an Benutzer

- Man kann Ergebnisse formatiert an den Benutzer zurückgeben.
- Beispiel:

```
age=21;
name='Hilde';
[s err] = sprintf('%s ist gestern %d Jahre alt geworden', name, age);
disp(s)
```

- Weitere Informationen: `doc sprintf`

Tiefergehende Grundlage

# Arbeiten mit Matrizen

# Erzeugen von Matrizen

- Wir wissen bereits, wie man Matrizen eingibt. MATLAB ® bietet aber auch die Möglichkeit bestimmt Matrizen über Funktionen zu definieren.

| Befehl | Matrix |
|---|---|
| `zeros(n,m)` | n x m Matrix, die nur aus Nullen besteht |
| `ones(n,m)` | Matrix aus Einsen |
| `rand(n,m)` | gleichmäßig verteilte zufällige Elemente |
| `randn(n,m)` | normal verteilte zufällige Elemente |
| `eye(n)` | n x n Einheitsmatrix |

# Einlesen von Matrizen

- Matrizen können auch aus externen Dateien eingelesen werden.
  - Beispiel 1: *.dat files

magic.dat:

```
16.0 3.0 2.0 13.0
5.0 10.0 11.0 8.0
9.0 6.0 7.0 12.0
4.0 15.0 14.0 1.0
```

```
>> load magic.dat
```

- Beispiel 2: M-Files

  File -> New -> Script

```
A = […
        16.0 3.0 2.0 13.0
        5.0 10.0 11.0 8.0
        9.0 6.0 7.0 12.0
        4.0 15.0 14.0 1.0];
```

  Save as -> magic.m

  Type: magic

# Zusammenfügen von Matrizen

- MATLAB ® bietet die sehr einfache Möglichkeit vorhandene Matrizen zu einer Größeren zusammen zu fügen.
- Nehmen wir zum Beispiel unsere bekannte Matrix `A`:

```
B = [A A+32; A+48 A+16]
```

- Dies erzeugt eine neue Matrix, die aus vier Untermatrizen zusammengesetzt wurde.

# Löschen von Zeilen und Spalten

- Zeilen und Spalten können in einer Matrix auch sehr einfach gelöscht werden:

```
X = A;
X(:,2) = [] %das löscht die zweite Spalte
```

Aufgabe: Löscht die zweite Zeile der Matrix A und speichert das Ergebnis in eine Matrix Y

Erscheinungsbild ändern

# Anpassen des Command Fensters

# Anpassen der angezeigten Genauigkeit

- `format` Kommando:

```
x = [4/3 1.2345e-6]

format short

   1.3333    0.0000

format short e

   1.3333e+000   1.2345e-006

format short g

   1.3333   1.2345e-006

format long

   1.33333333333333    0.00000123450000

format long e

   1.333333333333333e+000    1.234500000000000e-006
```

```
format long g

   1.33333333333333                1.2345e-006

format bank

   1.33           0.00

format rat

   4/3            1/810045

format hex

   3ff5555555555555    3eb4b6231abfd271
```

# Ausgabe verhindern

- Wenn ein Ergebnis nicht ausgegeben werden soll, so muss man die Eingabezeile mit einem Semikolon beenden

```
A = magic(100);
```

# Lange Eingaben

- Wenn ein Befehl nicht in eine Zeile passt, so kann man … verwenden, um anzudeuten, dass der Befehl in der nächsten Zeile weiter geht.

```
s = 1 - 1/2 + 1/3 - 1/4 + 1/5 - 1/6 + 1/7 …
      - 1/8 + 1/9 - 1/10 + 1/11 - 1/12;
```

- Anmerkung: vorherige Eingaben kann man mit dem Pfeil nach oben erneut aufrufen.

Ergebnisse visualisieren

# Grafiken

# Die `plot` Funktion

- `plot` hat, abhängig von den Eingabeparametern, verschiedene Formen
  - Ist `y` ein Vektor, so liefert `plot(y)` einen stückweise linearen Graphen der Elemente von y gegen Ihre Indizes.
  - Sind `x,y` beides Vektoren, so liefert `plot(x,y)` einen Graphen von `x` gegen `y`

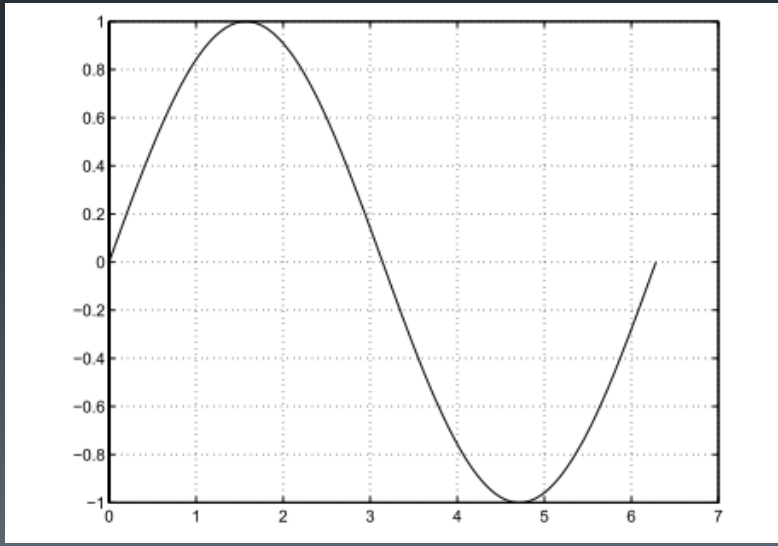
# Gitterpunkte zeichnen

- MATLAB ® zeichnet keine analytische Funktionen, wie etwa:

```
plot(sin(x),x,0,2*pi)
```

- Es müssen Gitternetze konstruiert werden
- Beispiel:

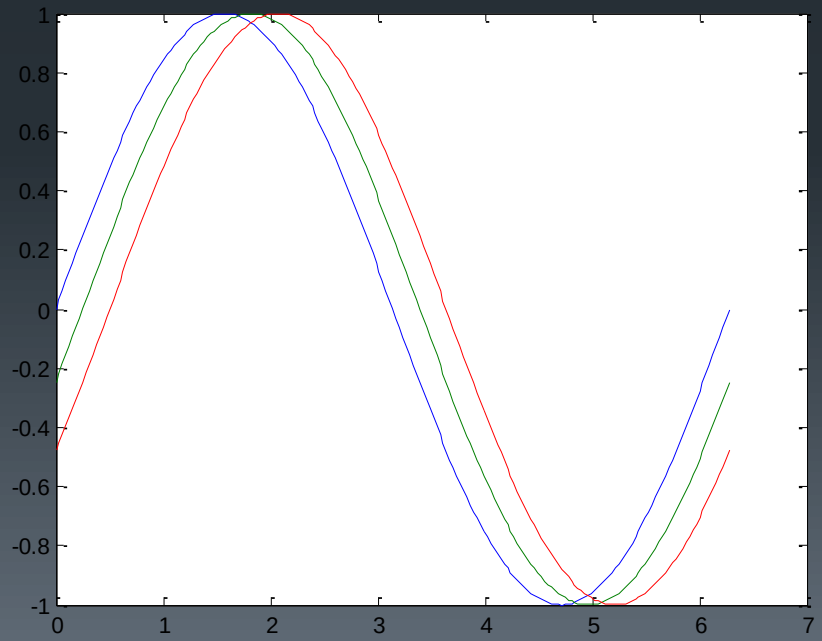
```
t=0:pi/100:2*pi;
y=sin(t);
plot(t,y)
```



# Mehrere Graphen in einen Plot

- Über gleichem Gitter
- Farben werden automatisch festgelegt

```
y2=sin(t-.25);
y3=sin(t-.5);
plot(t,y,t,y2,t,y3)
```



# Formatieren von Graphen

- Nach den Vektoren einen String mit bis zu drei Zeichen
- Farbstrings: c, m, y, r, g, b, w, k
- Linienart: -, --, :, -., none
- Markertypen: +, o, *, x

- Beispiel:

```
plot(x,y,'y:+')
```

- `Plot` zeichnet immer in ein Fenster, falls ein neues geöffnet werden soll -> `figure`

# In vorhanden Graphen neue Plots einfügen

- Wenn man in ein vorhandenes Figur-Fenster einen Plot hinzu zeichnen möchte, verwendet man den Befehl

  `hold on`

- Sobald man fertig ist, muss der Befehl wieder beendet werden, mit `hold off`

```
EDU>> [x y z]=peaks;
EDU>> contour(x,y,z,20,'k')
EDU>> hold on
EDU>> pcolor(x,y,z)
EDU>> shading interp
```

# Subplots

- Mehrere Plots in einem Fenster
- Fenster wird in m x n Matrix aufgeteilt
- `subplot(m,n,p)`

```
t = 0:pi/10:2*pi;
[X,Y,Z] = cylinder(4*cos(t));
subplot(2,2,1)
mesh(X)
subplot(2,2,2); mesh(Y)
subplot(2,2,3); mesh(Z)
subplot(2,2,4); mesh(X,Y,Z)
```

# Achsentitel und -formatierung

- `axis` liefert die Möglichkeit die Skalierung der Achsen zu bearbeiten bzw. sie ganz auszublenden
- `grid` schaltet das Gitternetz an und aus
- `xlabel, ylabel` und `zlabel` fügen Titel an die jeweilige Achse hinzu
- `title` setzt die Überschrift der Grafik
- `text` gibt die Möglichkeit überall im Plot einen Text zu plazieren

# Beispiel: Graphenformatierung

```
t = -pi:pi/100:pi;
y = sin(t);
plot(t,y)
axis([-pi pi -1 1])
xlabel('-\pi \leq \itt \leq \pi')
ylabel('sin(t)')
title('Graph der Sinusfunktion')
text(1,-1/3,'\it{Man bemerke die Symmetrie.}')
```

# Mesh- und Oberflächenplots

- `mesh` bietet ein Gitter
- `surf` zeichnet eine komplette Oberfläche
- Wird genutzt um Funktionen $f: \mathbb{R}^2 \longrightarrow \mathbb{R}$ zu visualisieren
- Wieder basierend auf einem Gitternetz
  - `meshgrid`

```
[X,Y] = meshgrid(-8:.5:8);
R = sqrt(X.^2 + Y.^2) + eps;
Z = sin(R)./R;
mesh(X,Y,Z)
```

# Bilder

- 2-dim Arrays können als Bilder interpretiert werden, wobei die Elemente des Arrays Helligkeit bzw. Farbe des Bildes bestimmen.
- Schönes Beispiel:

```
load durer
image(X)
colormap(map)
axis image
```

# Hilfe und Nachschlagewerke

**MATLAB ®**

- `help <function>`
- `doc <function>`
- `helpdesk`
- `lookfor <keyword>`
- `lookfor -all <keyword>`

http://www.tu-harburg.de/rzt/tuinfo/software/numsoft/matlab/kurse/einf/einf.html

**Octave**

- `help <function>`
- `doc <function>`
- `lookfor <keyword>`
- `lookfor -all <keyword>`

http://www.tu-harburg.de/rzt/tuinfo/software/numsoft/Einf_Octave.html

# Quellen

- Zwei Kurzeinführungen:
- http://www.tu-harburg.de/rzt/tuinfo/software/numsoft/matlab/kurse/einf/einf.html
- http://www.tu-harburg.de/rzt/tuinfo/software/numsoft/Einf_Octave.html

- MATLAB get started
- http://www.rzuser.uni-heidelberg.de/~jseyler/Files/getstart.pdf
- http://www.mathworks.com/help/techdoc/learn_matlab/bqr_2pl.html

- Zusätzliche Scripte
- http://www.esi.ac.at/~susanne/MatlabSkriptum.pdf
- http://rowicus.ch/Wir/Matlab_Octave/Matlab_Octave00.pdf

- Online Tutorial
- http://mo.mathematik.uni-stuttgart.de/kurse/kurs4/index_full.html