Einführung in Matlab

Matthias Kohl

RheinAhrCampus

Inhalt

1. Kapitel: Grundlagen
- Matlab-Oberfläche
- Daten als Vektoren und Matrizen
- for - und while Schleifen
- if - statement
- Darstellen von Daten
- Function (Unterprogramme)
- Schreiben und Lesen von Daten

2. Kapitel: Anwendungen und Beispiele
aus Medizintechnik und Sportwissenschaft

1. Kapitel: Grundlagen

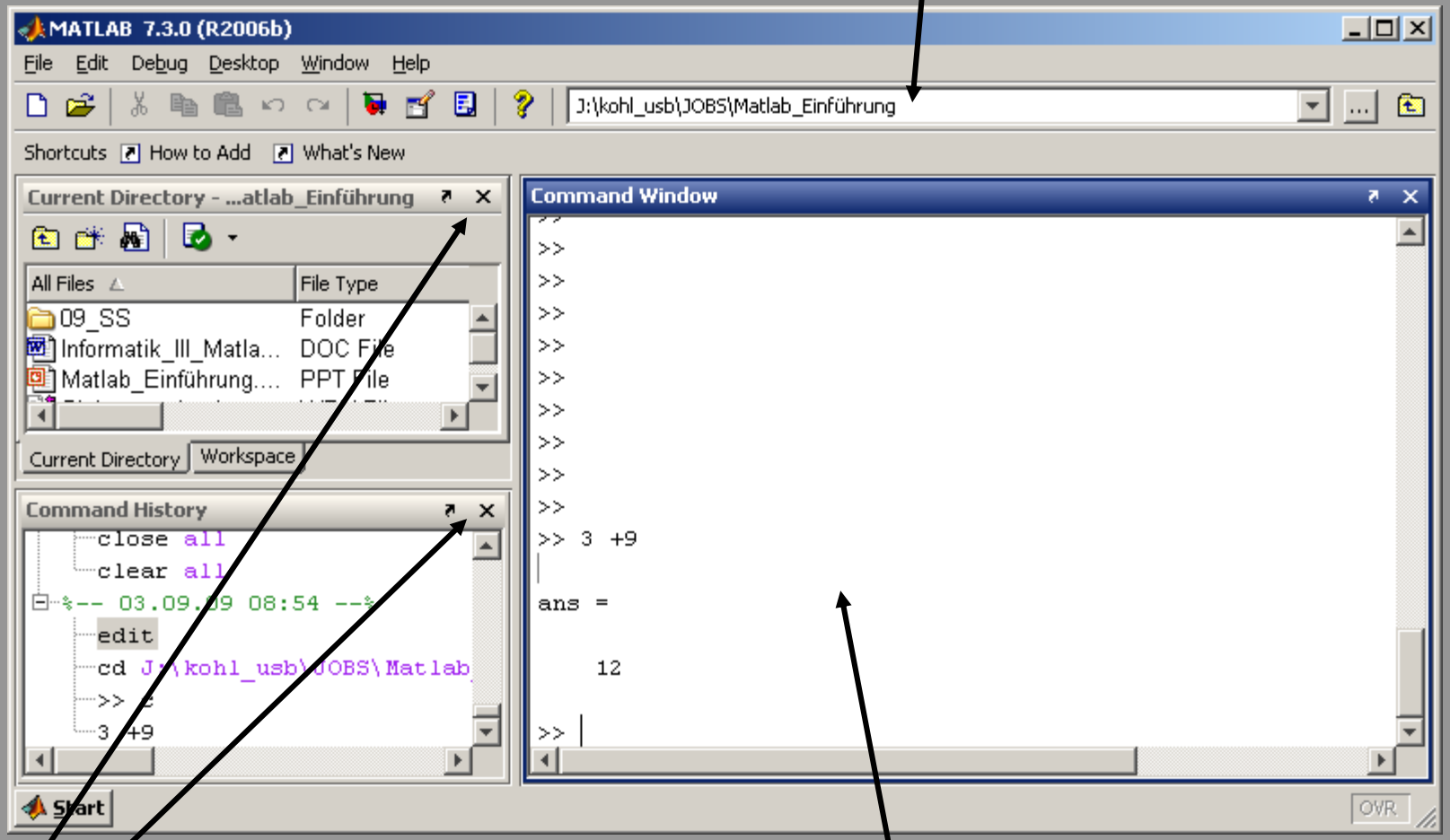
Freeware Alternativen zu Matlab

- Scilab: http://www.scilab.org
- Octave: http://www.octave.org
- Freemat: http://freemat.sourceforge.net/

- für einfache Programme kompatibel mit Matlab
- Standardbefehle
- nicht sehr umfangreich
- kostenlos!

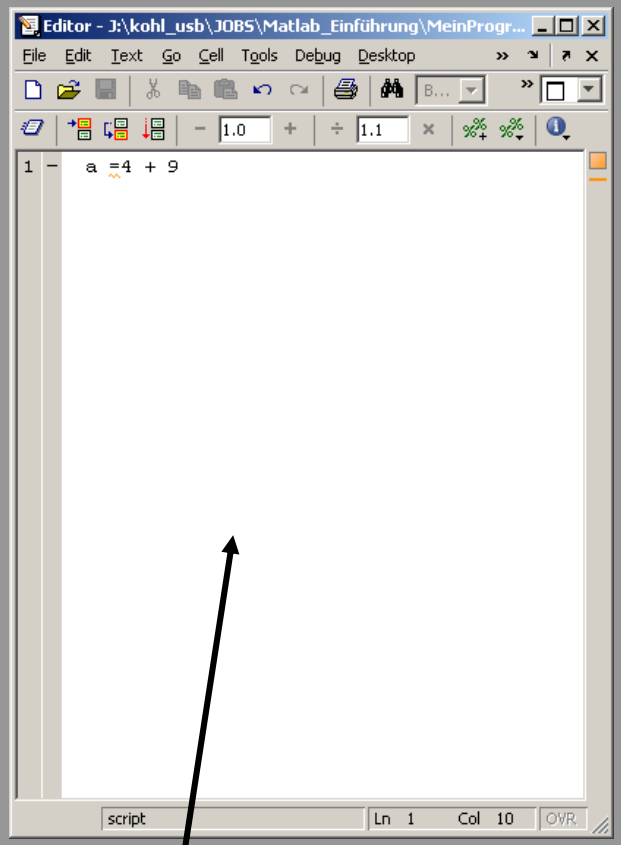
Matlab-Oberfläche

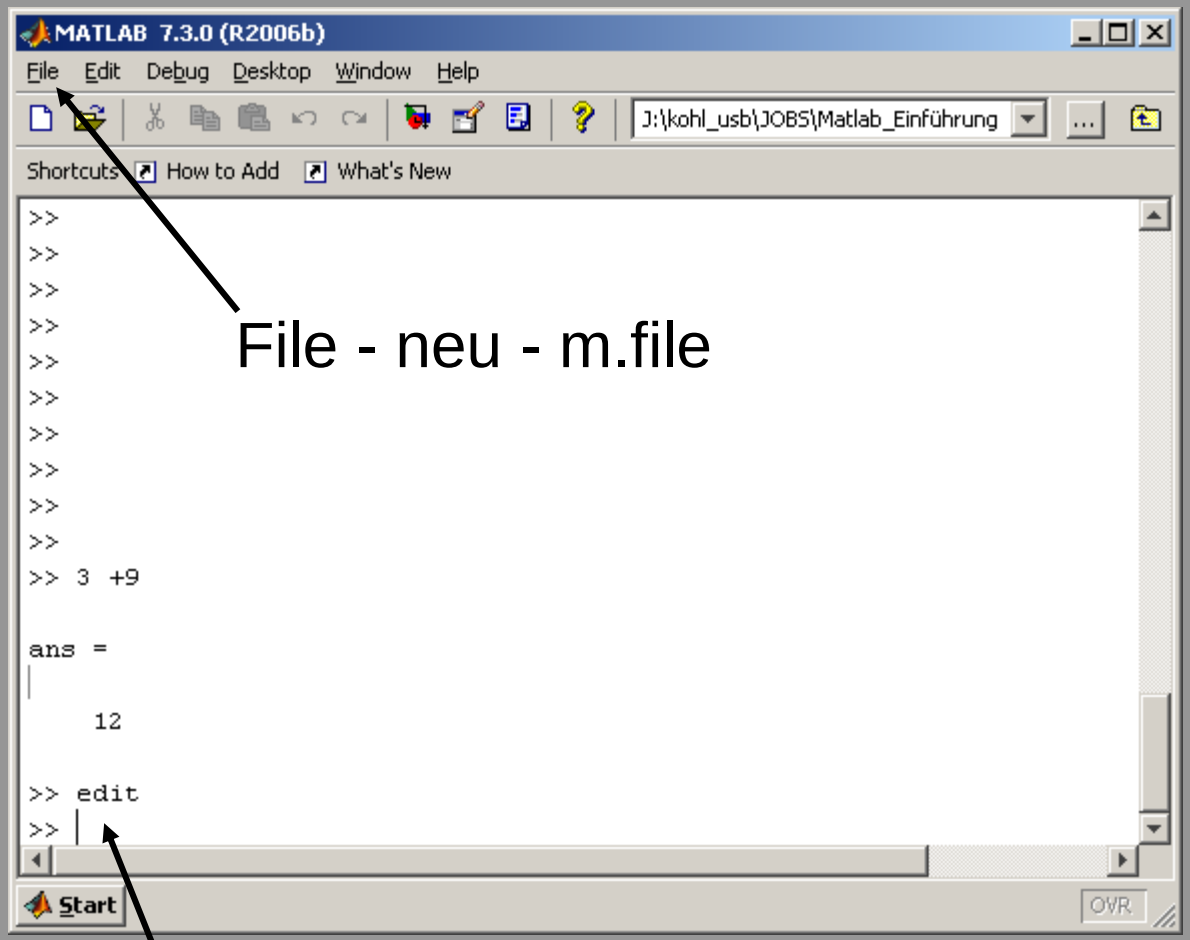
- nach dem Start



Matlab-Oberfläche

- Editor öffnen: *edit* - Editor window wird geöffnet





File - neu - m.file

Editor window

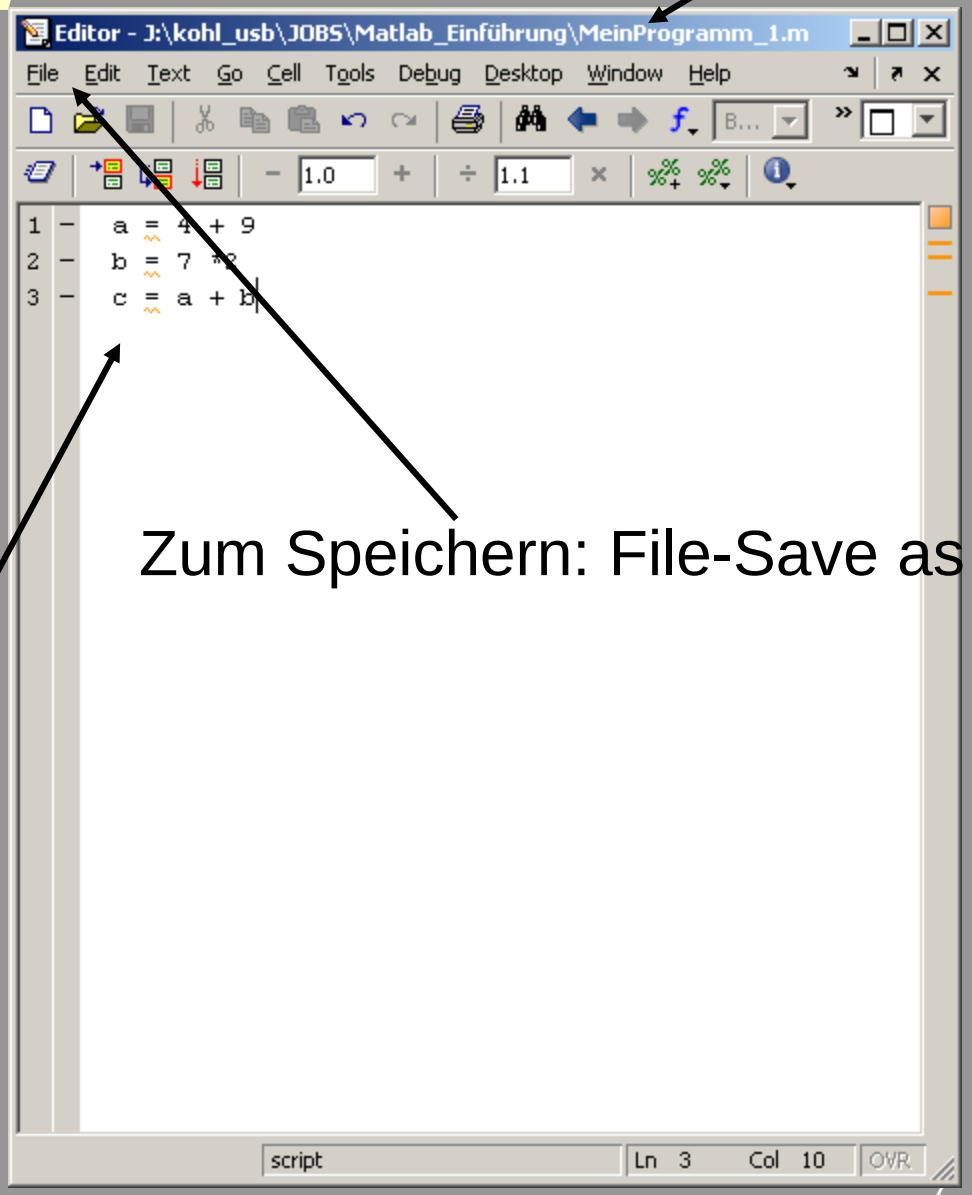
Eingabe „edit“ & return

Editor - Window

- um Programme zu schreiben / starten

Name:
- Endung „.m‘!
- keine Sonderzeichen verwenden
- auf path achten (Schreibberechtigung?)

Programm-Name & -Path



Zum Speichern: File-Save as

Zeilenorientierte Eingabe

Starten von Programmen

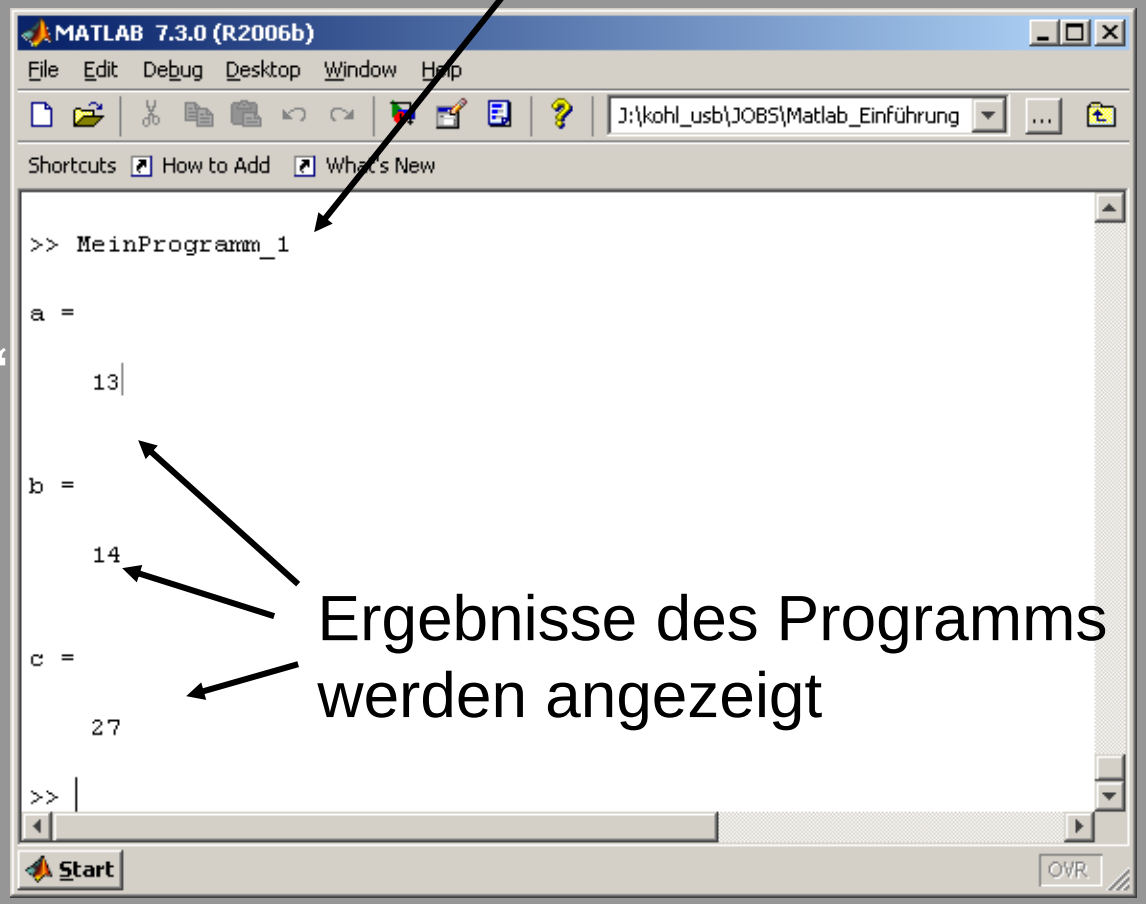
- Entweder:

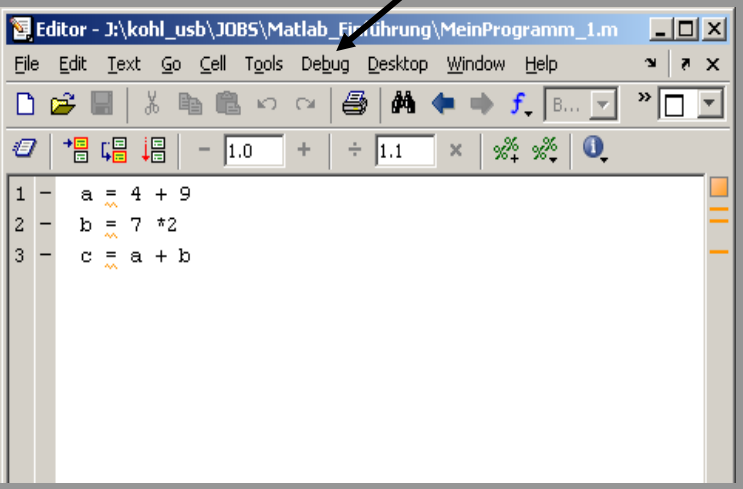
- im Command Window:
  Programmname

- Oder:

- Editor: ‚F5‘

- Editor: ‚debug run‘

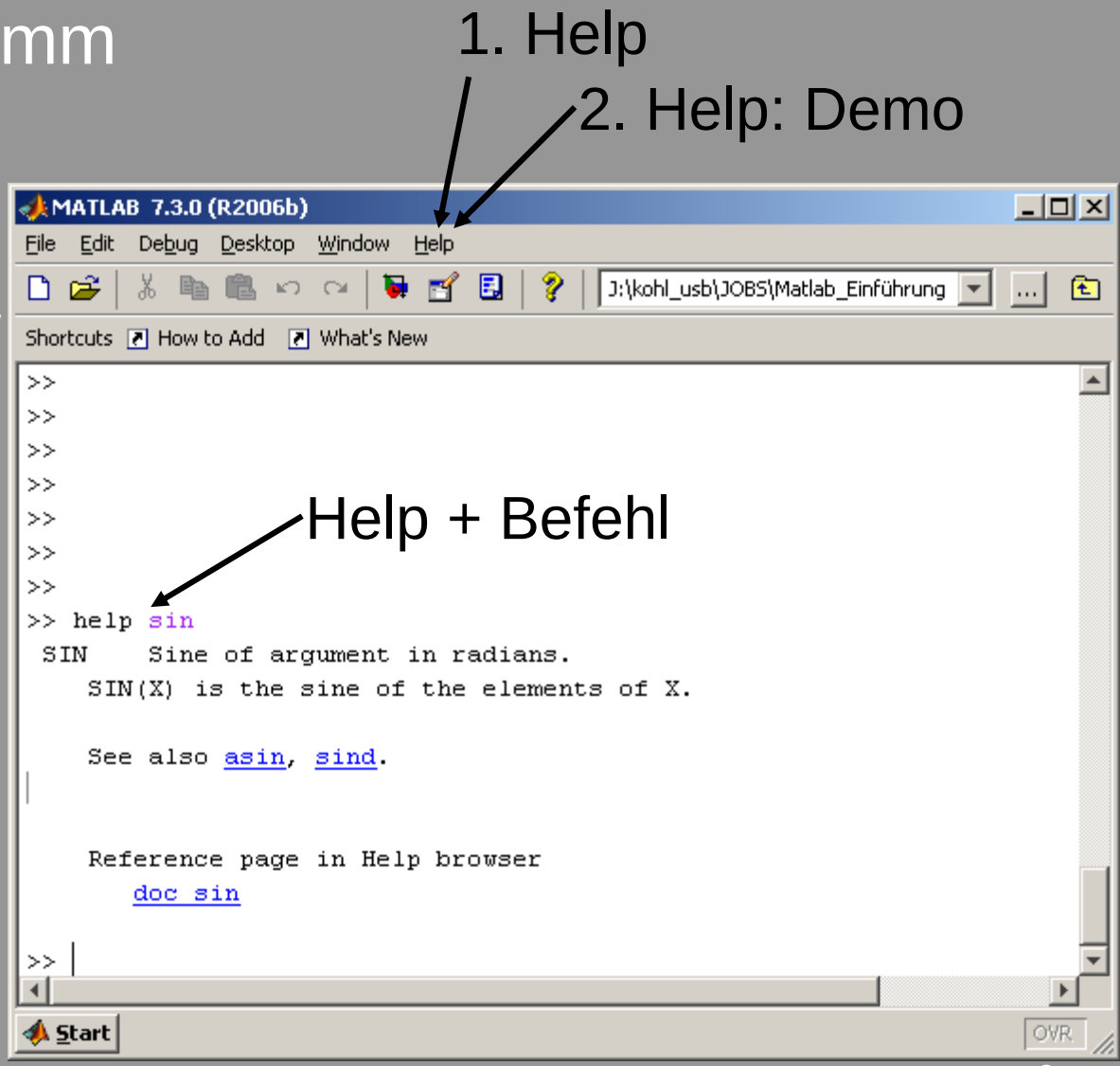




Hilfe!!!! ... in Matlab

1. Help - Full Programm
2. Help - Demos
3. ‚help' im command window
4. Homepage von Matlab:

www.mathworks.com



... häufige Fehler ...

1. Path nicht richtig gesetzt
2. Keine Schreibberechtigung für Path
3. m-Filenamen: möglichst keine Sonderzeichen; Endung „.m“; keine Zahl am Anfang
4. Dezimalstelle: „ . “ nicht „ , “
5. In Variablen und Filenamen leichte Verwechselung von ‚l‘ und ‚1‘, ... vermeiden!

... wir beginnen zu

Programmieren

For-Schleife

- Beispiel: Addition der Zahlen von 1 bis 4

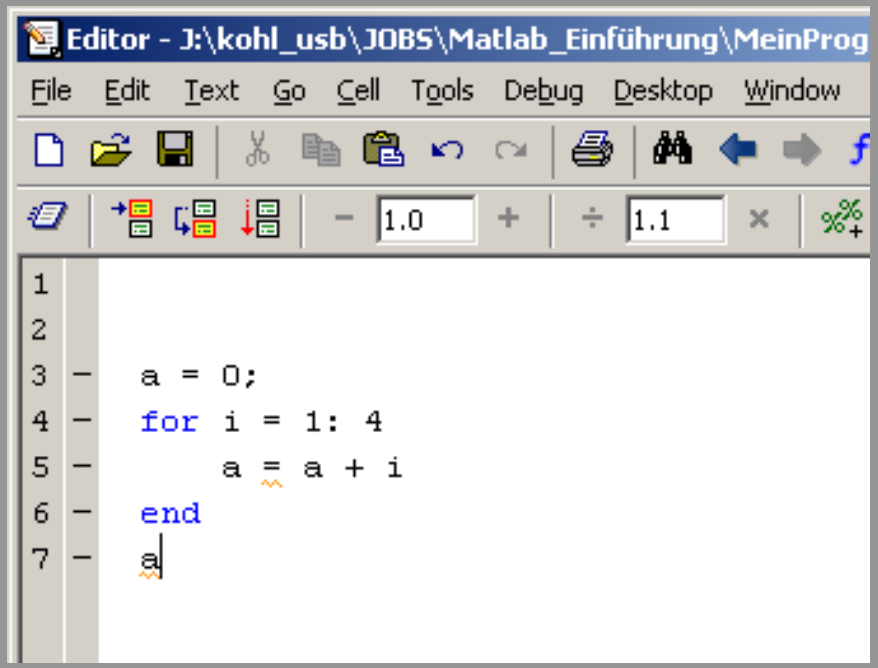
„der Laufparameter i läuft von 1 bis 4“

„alles zwischen „for“ und „end“ wird ausgeführt

„Anzeige des Ergebnisses“

```
a = 0;
for i = 1: 4
    a = a + i
end
a
```

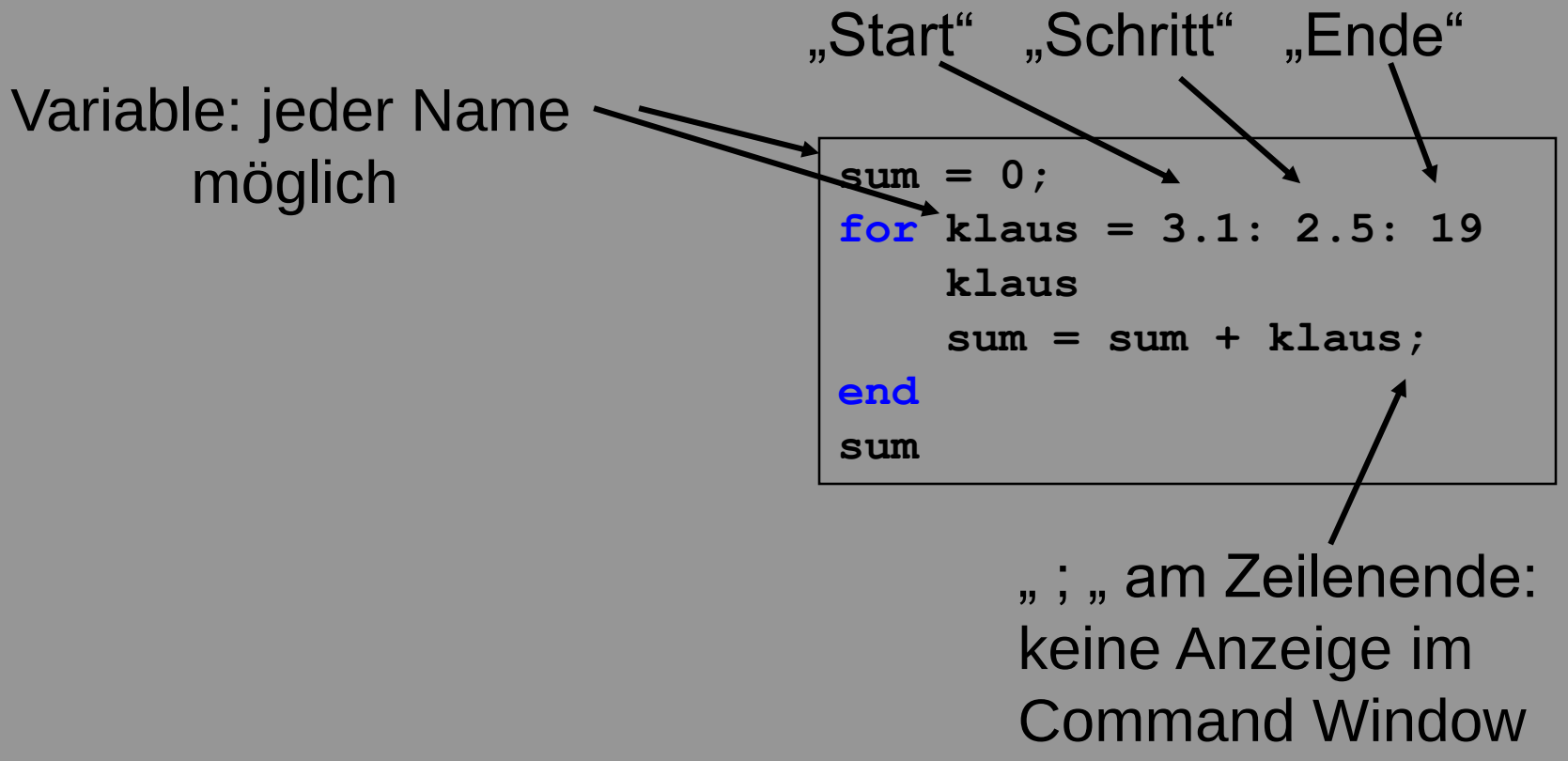
im Editor:



... dann Ausführen (Editor:„F5“) des Programms liefert „10“

For-Schleife: Hinweise

- Beispiel: Addition der Zahlen von 3.1 bis 19 in Schritten von 2.5
  (ab jetzt: Dezimalzahlen mit , . ‘!)



Daten als Vektoren (1D)

- Definition einer Variablen mit mehreren Elementen

```
v = [1 4 2 7 9 ]
```

Name

in [...]

- Aufrufen von Elementen

Zeige das 2. Element!

Zeige das 1. bis 5. Element mit Schrittweite 2! (wie bei for-Schleife)

```
V(2)   ¬

ans =

       4

V(1:2:5) ¬

ans =

       1 2 9
```

Daten als Vektoren (1D)

- Zeilenvektor
- Spaltenvektor

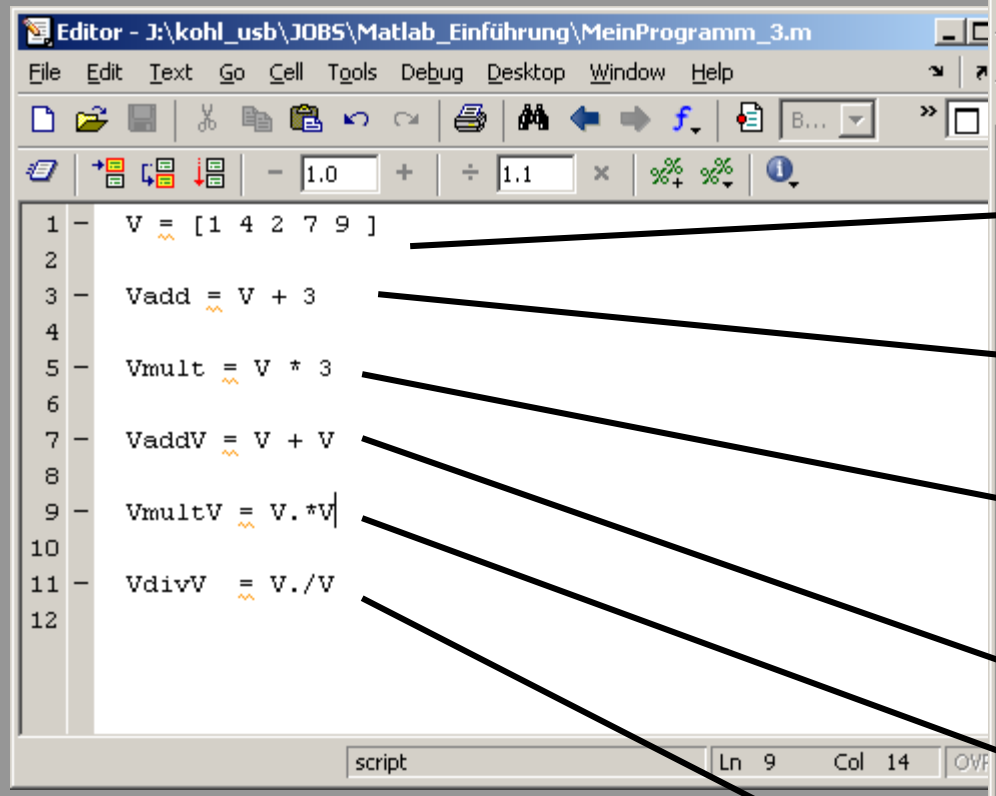
```
VZ = [1 4 2 7 9 ]
```

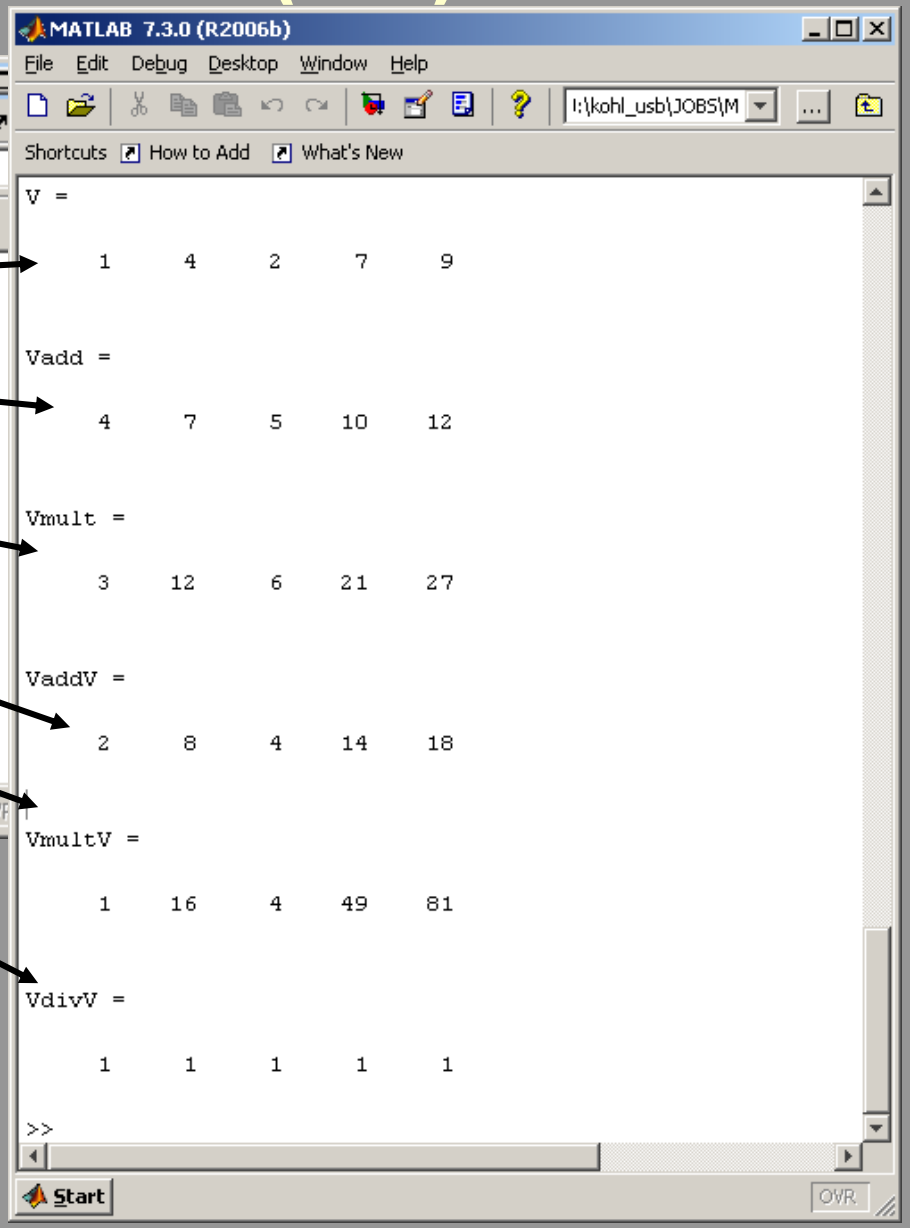
```
VS = [1; 4; 2; 7; 9 ]
```

- Abfragen der Länge bzw. der Größe

```
length(VZ)
length(VS)

size(VZ)
size(VS)
```

Rechnen mit Vektoren (1D)





Elementweise Multiplikation & Division: „ .* „ bzw. „ ./ „

Punkt!

Rechnen mit Vektoren (1D)

- gleiche Elementanzahl notwendig!!
  ... Fehlermeldung im Command-Window
- Unterschiedliche Kombinationen möglich

```
a = [1 2 3 4 5 ]
b = [6 5 4 3 2 1 ]

c = a - b

??? Error using ==> minus
Matrix dimensions must
agree.
```

Welche Werte haben die Variablen?

```
c = a(3) + b(6)
d = a - b(1:5)
e = a - b(5:-1:1)
f = a(2:4) - b(1:3)
g = a(1:2:5) - b(2:2:6)
```

- Index muß positiv und ganzzahlig sein!!

```
h(-3:1:-1)= [ 3 2 1]
m = a(3.3) + b(1.7)
```

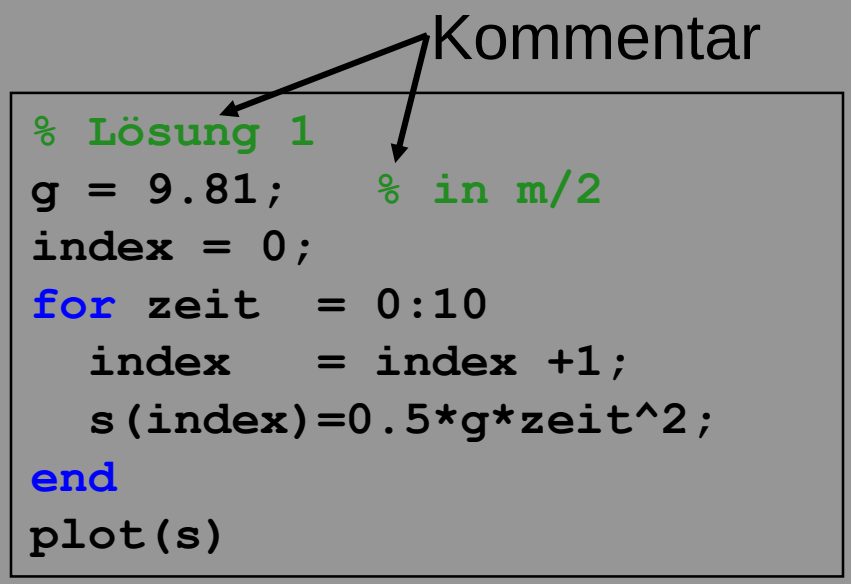
falsch - Fehlermeldungen erscheinen!!

Rechnen mit Vektoren: Beispiel 1

- Freier Fall: Berechnen Sie die Fallstrecke für Zeiten < 10 s!
- Ansatz: $s(t) = 1/2 \cdot g \cdot t^2$
- .... mit Zeit t

- Hier Lösung 1 und 2
- ... auf der nächsten Seite

  Lösung 3 bis 5




```
% Lösung 2
g = 9.81;   % in m/2
for index =1:100
    zeit  =index/10;
    s(index) =0.5*g*zeit^2;
end
plot(s)
```

Rechnen mit Vektoren: Beispiel 1

- Freier Fall: Berechnen Sie die Fallstrecke für Zeiten < 10 s!

Kommentar

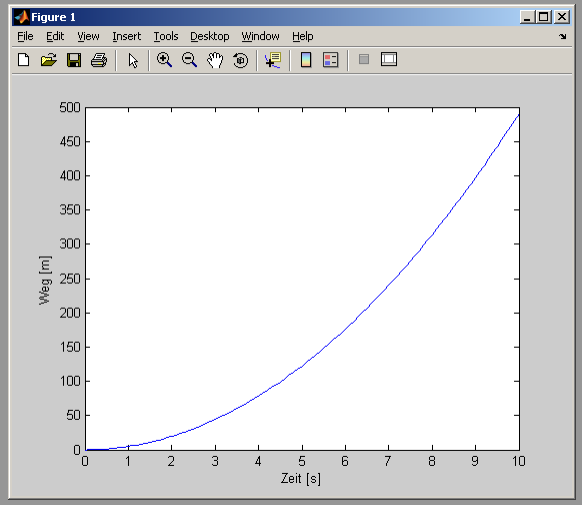
```
% Lösung 3
g = 9.81;    % in m/2
for index =1:100
 zeit(index)=index/10;
 s(index)=0.5*g*zeit(index)^2;
end
plot(zeit,s)
```

```
% Lösung 4
g = 9.81;    % in m/2
for index =1:100
    zeit(index)  =index/10;
end
s = 0.5*g*zeit.^2;
plot(zeit,s)
```

ohne for-Schleife

Achsen-Beschriftung

```
% Lösung 5
g = 9.81;    % in m/2
zeit =[0:0.1:10];
s = 0.5*g*zeit.^2;
plot(zeit,s)
xlabel('Zeit [s]')
ylabel('Weg [m]')
```



While-Schleife

- Beispiel: Addition der Zahlen von 0 bis 100 in Schritten von 0.1

Vergleichsoperator:

„ == <, <=, >, >= „

```
Zaehler = 0;
Schritt = 0.1;
Sum     = 0;
while Zaehler < 100
   Sum = Sum + Zaehler;
   Zaehler= Zaehler+Schritt;
end
Zaehler
```

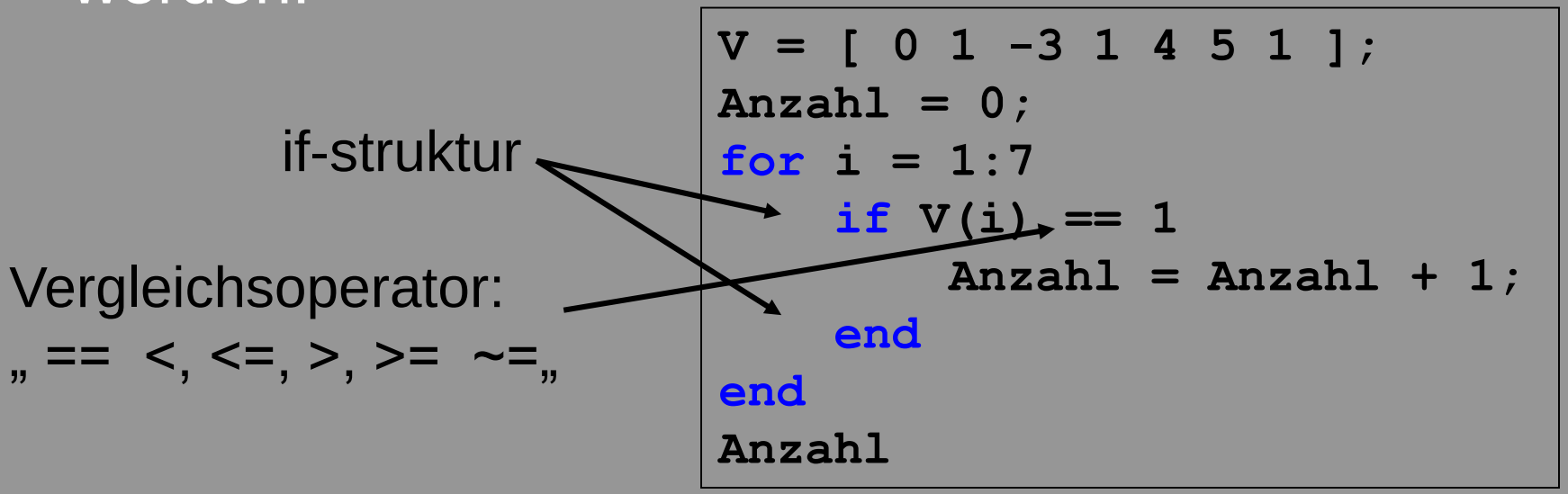
Variable wird verändert!!!

… Gefahr: Endlosschleife, wenn Abbruch nie erfüllt!!!

… zum Unterbrechen eines Programmes im Command-Window: „Strg C“.

If-Struktur

- Beispiel: Aus einem gegebenen Vektor soll die Anzahl der Elemente mit dem Wert „1“ gefunden werden!



Achtung: logisches „gleich“: „==“

Aufgabe: Kleine Statistik

Kleine Statistik: Untersuchung eines Vektors

Gegeben sein ein beliebiger Vektor, z. B. W = [1 3.2 1.2 4.6 4.3 9 3].

Aufgabe: Verwenden Sie for- oder while-loops, um das Minimum, das Maximum, den Mittelwert sowie die Standardabweichung zu berechnen. Diese vier Werte sollen vier Elemente eines neuen Vektors „Ergebnis“ sein.

Der Mittelwerte ist definiert als $< n >= \frac{1}{N}\sum_{i=1}^{N} n_i$, wobei n_i die Einzelwerte sind und N die Anzahl dieser Werte. Die Standardabweichung ist definiert als:

$$\sigma = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(n_i - < n >)^2} \; .$$

Hinweis: In Matlab gibt es bereits fertige Befehle für die Berechnung des Mittelwerts („mean“), der Standardabweichung („std“) sowie „min“ und „max“; diese können zur Überprüfung der Ergebnisse verwenden werden. Der Befehl für die Quadratwurzel ist „sqrt“.

Aufgabe: Kleine Statistik - Lösung

```
W = [ 1  3.2  1.2  4.6  4.3  9   3];
Maximum = W(1); Minimum = W(1);
Mittelwert = 0; Standardabw=0;

for i = 1: length(W)
   if W(i) > Maximum
      Maximum = W(i);
   end
   if W(i) < Minimum
      Minimum = W(i);
   end
   Mittelwert = Mittelwert +W(i);
end
Mittelwert = Mittelwert/length(W);

for i = 1:length(W)
   Standardabw = Standardabw + (W(i) - Mittelwert)^2;
end
Standardabw = sqrt(Standardabw/(length(W)-1));
```

Aufgabe: Datenvektor

Erstellen eines Datenvektors

Aufgabe:

a) Verwenden Sie eine for-loop, um einen Datenvektor V = [0 2 4 6 8 10 ... 20] zu erstellen!

b) Verwenden Sie eine for-loop, um einen Datenvektor V = [-1 +4 -9 +16 ... 100] zu erstellen!

c) Erstellen Sie die Vektoren aus a) und b) ohne Verwendung einer for - loop!

d) Ändern Sie die Programme aus a) und b) so, dass die Summe der Elemente berechnet wird!

e) Stellen Sie die Vektoren graphisch dar!

Aufgabe: Datenvektor - Lösung

```
% Teil a)
clear all  % damit alle Variablen gelöscht werden
for i = 1: 11
   V(i)=(i - 1) * 2;
   V1(i)=i*2 - 2;  % andere Schreibweise
end
V
V1

% Teil b)
for i = 1:10
   V2(i)= (-1)^i*i^2;
end
V2

% Teil c)
V3 = [0:2:20]
% oder
V4 = [1:11] * 2 - 2
V4 = (-1).^[1:10].*[1:10].^2
```

```
% Teil d)
Summe = 0;
for i = 1: 11
   V(i)=(i - 1) * 2;
   V1(i)=i*2 - 2;  % andere Schreibweise
   Summe = Summe + V(i);
end
V
V1
Summe
% Teil e)

plot(V)
hold on
plot(V4)
```

Daten als Matrixen (2D)

- Definition: 2D-Variablen ergibt:

```
A =
    0    1    2
    4    5    6
    3    1    4
    5    4    3
```

neue Zeile

```
A= [0 1 2 ;4 5 6;3 1 4;5 4 3 ]

size(A)
```

- Größe einer Matrix

ans = 4 3

- Rechnen mit Matrizen

```
B = A - 3

C = A.*B
```

Elementweise Operation: „ . “

Daten als Matrixen (2D)

- Teilbereiche einer Matrix können auf unterschiedliche Weise ausgeschnitten werden

```
A= [0 1 2 ;4 5 6;3 1 4;5 4 3 ];
A =
     0     1     2
     4     5     6
     3     1     4
     5     4     3

A(1,1:3)
ans =
     0     1     2
A(4,2:3)
ans =
     4     3

A(3,:)
ans =
     3     1     4

A(2,3:-1:1)
ans =
     6     5     4
```

Daten als Matrixen (2D)

- Rechnen mit Matrixen

```
A= [0 1 2 ;4 5 6;3 1 4;5 4 3 ];
```

Transponieren

```
B = transpose(A)
C = A'
```

Inverse

```
D = inv(A)
```

(nur bei quadratischen Matrizen)

```
sin(A)
```

elementweise

```
mean(A,1)
```

spaltenweise

```
mean(A,2)
```

zeilenweise

Aufgabe: Erstellen einer Matrix

Aufgabe:
a) Verwenden Sie zwei for-loops, um eine 4 x 4 Matrix zu erstellen, deren Elemente den Wert 3 haben!

b) Verwenden Sie zwei for-loops, um eine 4 x 5 Matrix

$$M = \begin{matrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 3 & 4 & 5 & 6 \\ 3 & 4 & 5 & 6 & 7 \\ 4 & 5 & 6 & 7 & 8 \end{matrix}$$

zu erstellen!

c) Ändern Sie die Programme so, dass die Summe der Elemente berechnet wird!

d) Stellen Sie die 2. und 4. Reihe graphisch dar!

Aufgabe: Erstellen einer Matrix - Lösung

```
% Teil a)
clear all
for i = 1: 4
   for j = 1: 4
      M1(i,j) = 3;
   end
end
M1

% Teil b)
for k = 1: 4
   for m = 1: 5
      M2(k,m)= k + (m-1);
   end
end
M2

% Teil c)
Summe = 0;
for k = 1: 4
   for m = 1: 5
      M2(k,m)= k + (m-1);
      Summe = Summe + M2(k,m);
end
end
M2
Summe
```

```
% Teil d)
figure %eine neue Figure wird erstellt
plot(M2(2,:),'-+k')
hold on
plot(M2(4,:),'-*r')
```

Vektoren und Matrixen: Hinweise

Automatisches Auffüllen mit ‚0‘

```
clear all
M(3,5)= 17
```

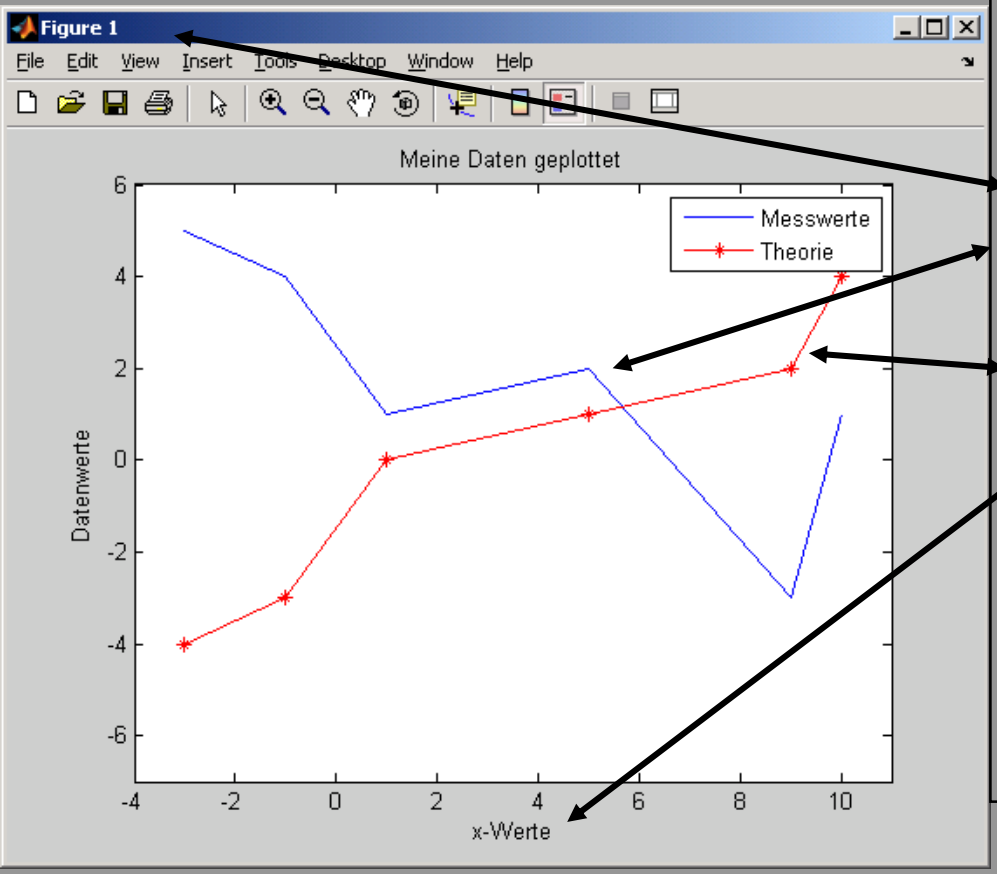


```
M =

     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0    17
```

```
V = [3 17 6 9 7];
V(end)

V(end-2)

V(end+3) = 1
```

```
ans =
     7
ans =
     6
V =
  3  17   6   9   7   0   0   1
```

Plot – Darstellung von Daten: 1



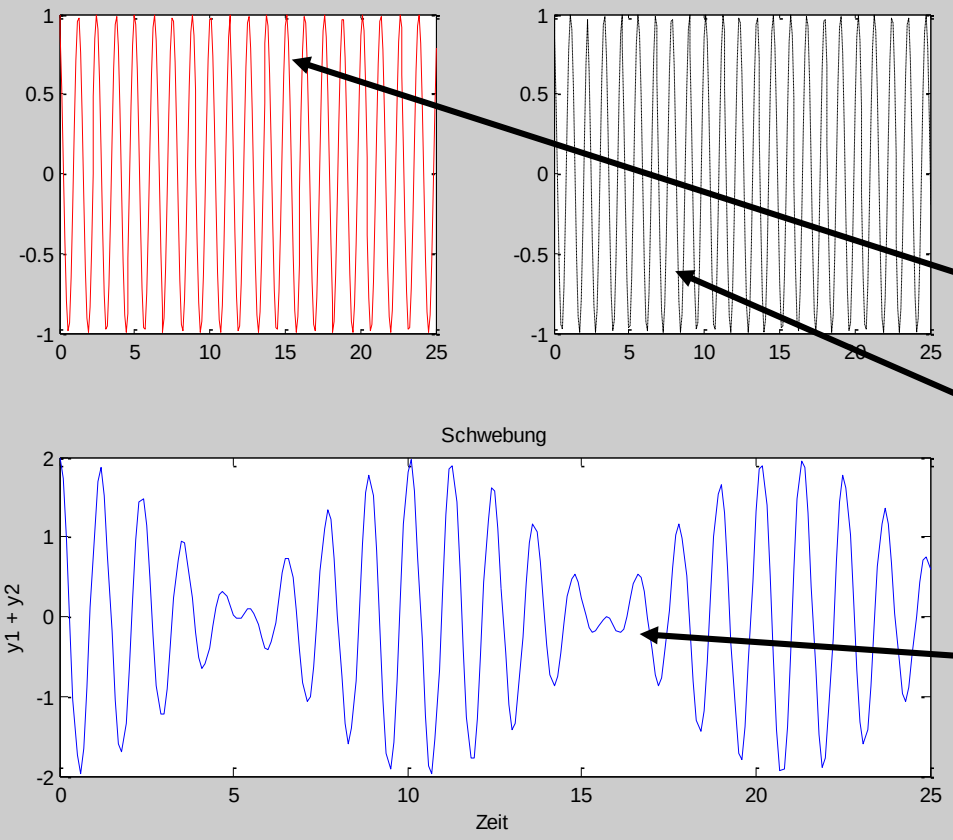

```
x = [ - 3 -1 1 5 9 10];
y = [ 5 4 1 2 -3 1 ];
z = [ -4 -3 0 1 2 4 ];

figure(1)
plot(x,y)
hold on
plot(x,z,'-*r')
hold off
xlabel('x-Werte')
ylabel('Datenwerte')
title('Meine Daten geplottet')
legend('Messwerte','Theorie')
axis([-4 11 -7 6])
```

Symbol & Farbe

Wertebereich

Plot – Darstellung von Daten: 2

Überlagerung von Schwingungen




```
% Überlagerung von
% Schwingungen
clear all
t = [0:0.1:25]; % Zeit
frequenz1= 5;
frequenz2= 5.6;
y1 =cos(frequenz1*t);
y2 =cos(frequenz2*t);

subplot(2,2,1)
plot(t,y1,'r-')

subplot(2,2,2)
plot(t,y2,'k--')

subplot(2,1,2)
plot(t,y1+y2)
xlabel('Zeit')
ylabel('y1 + y2')
title('Schwebung')
```

Plot – Darstellung von Daten: 3

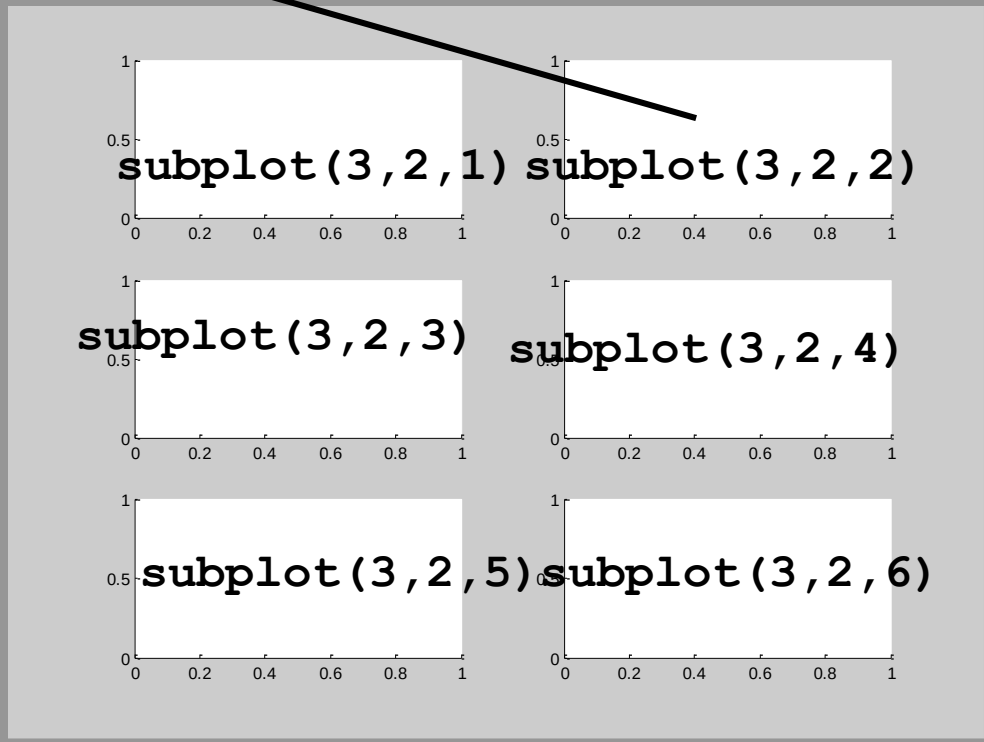
Details unter ‚help plot & subplot‘

```
subplot(3,2,2)
plot(t,y2,'xk--')
```

```
r red
g green
b blue
c cyan
m magenta
y yellow
k black
```

```
. Point
o circle
x x-mark
etc.
```

```
- solid
: dotted
-- dashed
etc.
```



Aufgabe: Plotten von Daten

Aufgabe:
a) Stellen Sie die zwei Funktionen $y = x^{1.4} - 2x - 0.5$ und $z = -x/3$ im Wertebereich zwischen 0 und 5 dar. Beschriften Sie die Achsen und geben Sie der Abbildung einen Titel, wobei durch Farben die beiden Funktionen kenntlich gemacht werden sollen.
b) Unterteilen Sie die Figur in zwei Bereiche („subplot“) und tragen Sie in den zweiten Bereich die Differenz y – z gegen x auf. Beschriften Sie die Achsen der Abbildungen!
c) Schätzen Sie den (numerischen) Schnittpunkt von y und z! Dazu kann der Wert der Differenz y – z bestimmt werden, der am nächsten an der Null liegt (dazu am besten den Betrag (Absolutwert) verwenden: „abs“). Der zugehörige x-Wert gibt den Schnittpunkt an.
Hinweis: Zum Erstellen wählen Sie eine Schrittweite Δx vor, z. B. 0.01. Verwenden Sie eine for- oder while-loop oder eine Vektorberechnung ohne loop.

Aufgabe: Plotten von Daten - Lösung

```
clear all
% Erste Möglichkeit zum Erstellen von y
x = [0:0.001:5];
y = x.^1.4 - 2*x -0.5;
z = -x/3;

% ebenso möglich mit for-loop
x = [];
y = [];
for i = 0:0.001:5
   x(end + 1)=i;
   y(end + 1) = i^1.4 - 2*i -0.5;
end
```

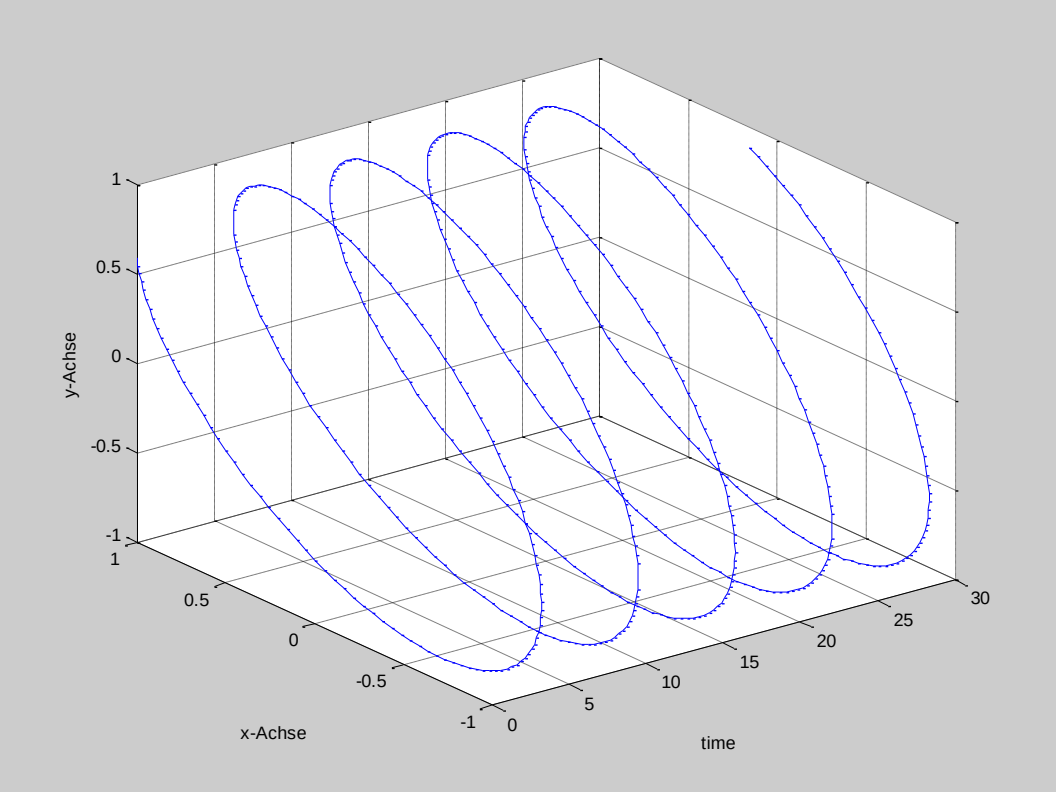
```
figure(1); subplot(2,1,1)
plot(x,y)
grid on ; hold on
plot(x,z,'r')
hold off
xlabel('x-Achse'); ylabel('y und z ')
title('Funktionswerte')
legend ('F1: y = x^1.4 - 2x - 0.5', 'F2: z = -
x/3')

subplot(2,1,2)
plot(x,y- z,'r')
grid on
xlabel('x-Achse')
ylabel('y-z ')
title('Differenz der Funktionen')
legend('y - z')

Delta = 1000;
for i = 1:length(x)
   if abs(y(i)-z(i)) < Delta
      Delta = abs(y(i)-z(i));
      Position = i;
   end
end
Schnittpunkt = x(Position)
```

Plot – Darstellung von Daten: 3D

Beispiel: Überlagerung von Schwingungen
in x- und y-Richtung mit Phasenverschiebung

```
% z. B. Polarisiertes Licht
clear all
t = [0:0.05:10];
E0 = 1;
phase = pi/1*0.3;
omega = 1;
Ex = E0*cos(omega * t);
Ey = E0*cos(omega * t+ phase);

figure(3)
plot3(t,Ex,Ey)
xlabel('time')
ylabel('x-Achse')
zlabel('y-Achse')
grid on
```

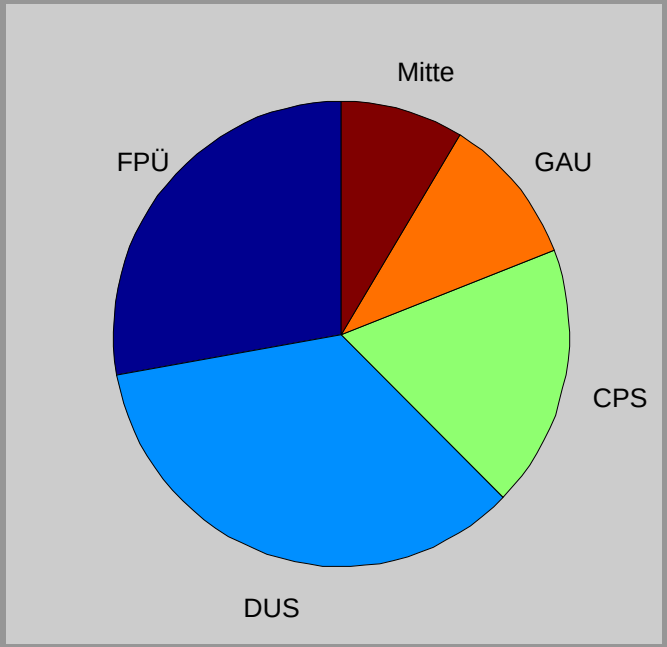
dies ist alles!!



Weitere Darstellungen von Daten

Pie

```
wahl = [ 32 40 21 12 10 ]
pie(wahl,{'FPÜ','DUS','CPS','GAU','Mitte'})
```



Einige wichtige Befehle 1

Erklärungen zur Benutzung aller Befehle: help
z. B. „help pwd“ im Command-Window

Basisbefehle

- clear (clear all)
- close all
- cd
- whos
- pwd („print working directory“)
- help
- break
- pause

Abbildungen / Darstellugn

- plot / bar / errorbar
- figure
- subplot
- hold on / hold off
- xlabel / ylabel
- legend / text
- imagesc

Zur Untersuchung von Daten

- length
- size
- find
- min / max
- round / floor

Dateien-Lesen / -Schreiben

- load
- save
- textread
- fscanf / fprintf
- fread

Einige wichtige Befehle 2

Statistik
- min / max
- mean / std / median / var
- corrcoef / cov
- ttest, ttest2, ztest

Trigonometrische Funktionen
- sin cos tan cot (‚rad‘)
- sind cosd tand cotd (‚degree‘)
- asin acos atan acot

Vorbestimmte Variablen
- pi
- i = sqrt(-1) (imaginäre Einheit)

Und anderes
- sqrt
- round / floor

Exp / log
- log („= log Basis e“)
- log10
- e = exp(1) („Eulersche Zahl“)

Aber: Schreibeweise von Zahlen

Die Zahl $1{,}3 \cdot 10^{17}$ wird geschrieben als ‚1.3e17‘ !!

(auch richtig: ‚1.3*10^17‘, aber dies enthält eine Rechenoperation)

Polynomfit

Polynomfit durch Daten

- ‚polyfit‘

Polynom-Wert können mit

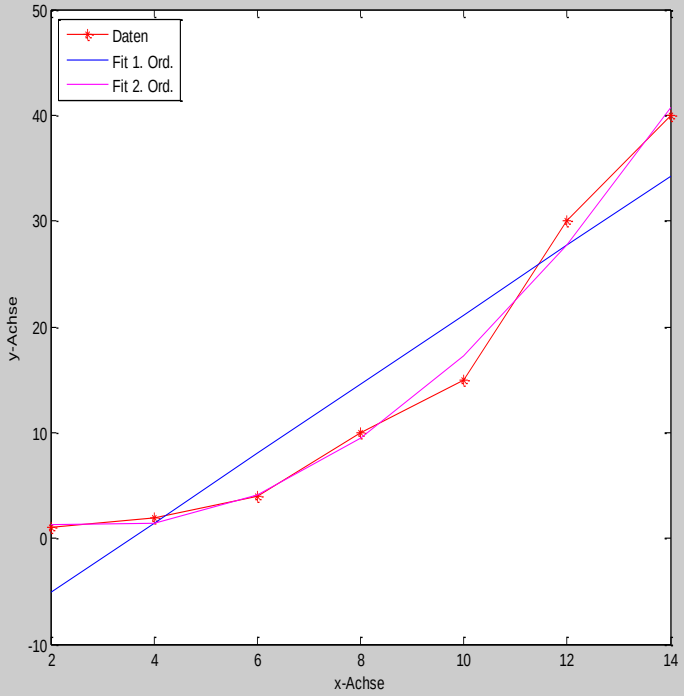
- ‚polyval‘

Bestimmt werden.
Hier ein Beispiel:

Polynom 1. Ordnung

Polynom 2. Ordnung

```
clear all
x = [2 4 6 8 10 12 14];
y = [ 1 2 4 10 15 30 40];

p1 = polyfit(x,y,1)
y_fit1= polyval(p1,x);

p2 = polyfit(x,y,2)
y_fit2= polyval(p2,x);

plot(x,y,'r-*')
hold on
plot(x,y_fit1,'b')
plot(x,y_fit2,'m')
hold off
legend('Daten','Fit 1. Ord.','Fit 2. Ord.',2)
xlabel('x-Achse')
ylabel('y-Achse')
```



Alternative:
In der Figure:
‚Tools/ Basic Fitting‘

Functions (Unterprogramme)

Functions
- Erstellen von Unterprogrammen

Unterprogramme (Subroutines) werden in Matlab „functions“ genannt.

Die Funktionsweise soll an einem einfachen Beispiel erläutert werden.

Aufgabe: eine Funktion soll von zwei beliebigen Zahlen die Summe und die Differenz berechnen.

Eingabewerte, hier zwei, getrennt durch „ , “ in ( )

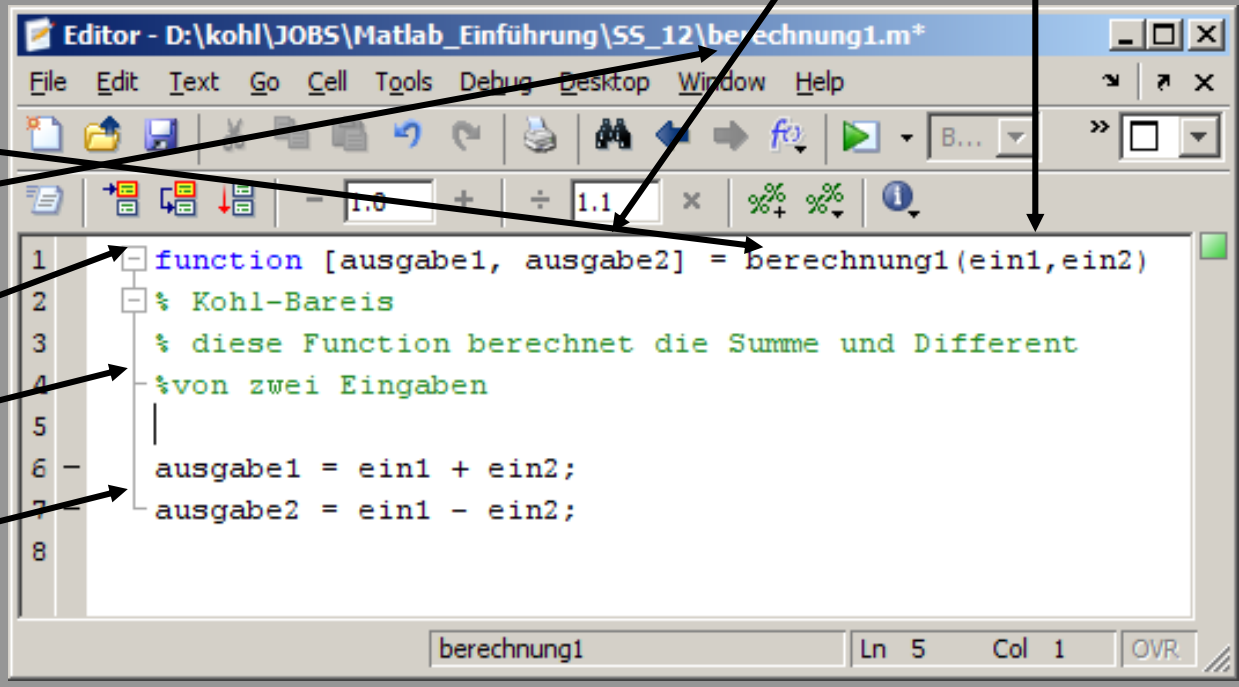
Ausgabewerte, hier zwei, getrennt durch „ , “ in [ ]

Name der function hinter =-Zeichen und auch so abgespeichert!

Erstes Wort: „function“

Kommentarzeilen

Berechnung der Ausgabewerte Aus den Eingabewerten



… auf der nächsten Seite: Aufruf der function

Functions
- Aufruf von Functions

Functions werden aus dem Command-Window oder einem anderen Programm aufgerufen

Beispiel: die Function „berechnung1.m“ soll für die Werte a = 3 und b = 5 aufgerufen werden

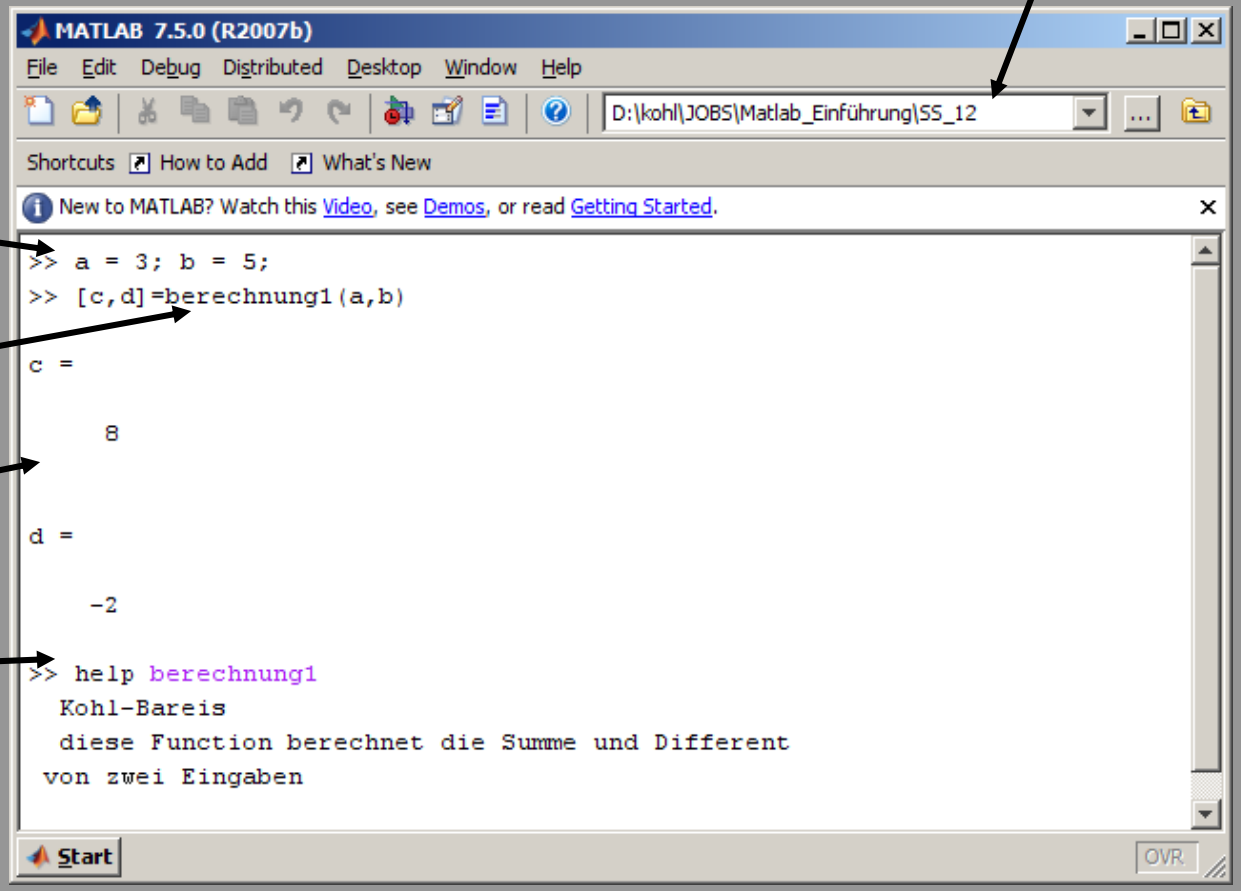
Achtung: die function muss unter der Directory liegen, die hier gewählt ist

Beliebige Variablen a & b

Aufruf der function mit a und b

c und d sind die neuen Ausgabewerte

durch help werden die Kommentarzeilen ausgegeben



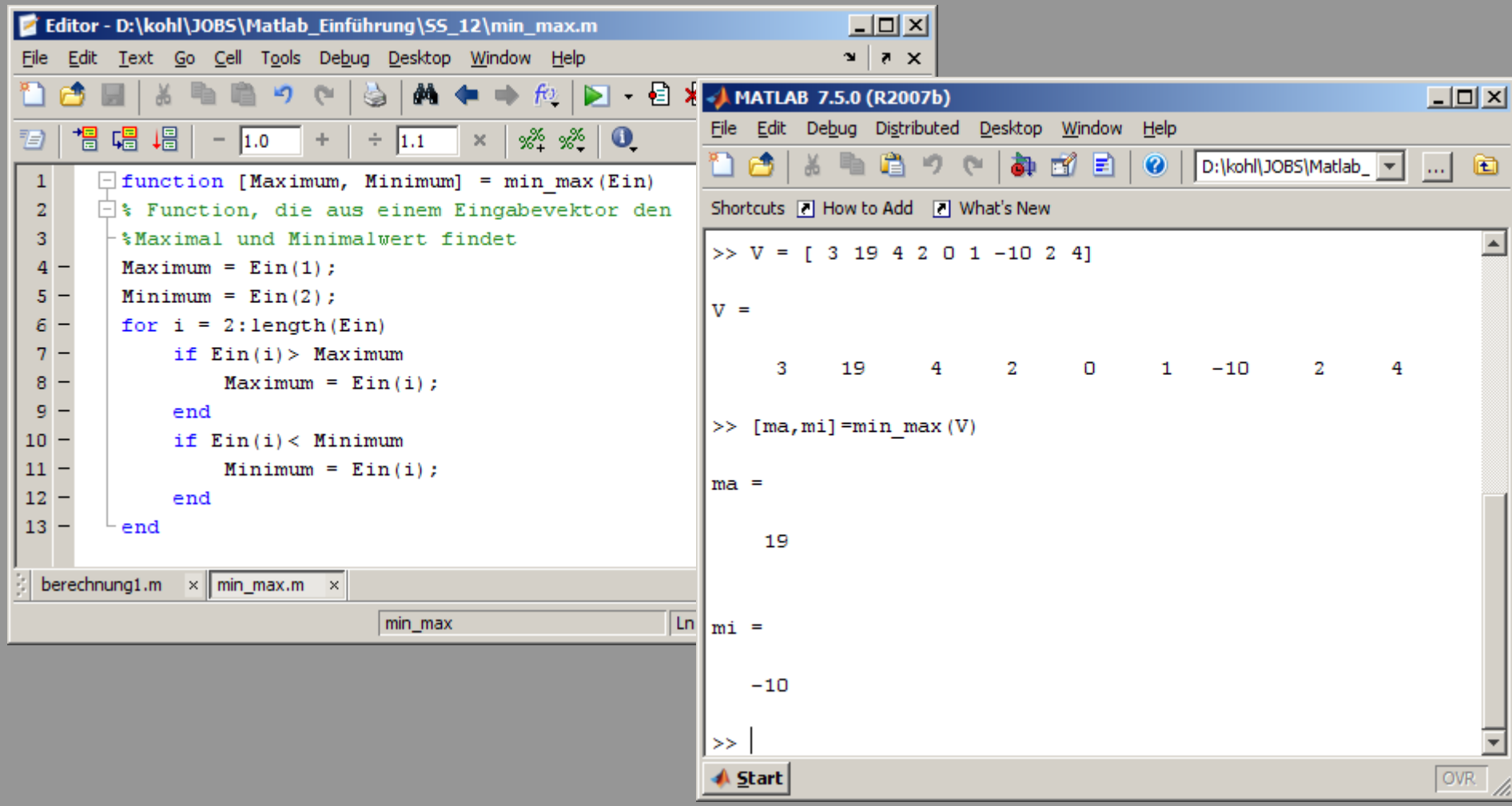
Functions - weiteres

Ein- und Ausgabeparameter können einzelne Werte, Vektoren oder Matrizen sein

Die Programmstruktur ist wie bei jedem anderen m-File

Achtung: eine function ist selbst nicht lauffähig, d.h. liefert einen Fehler beim Drücken von F5; die function muss aufgerufen werden

Functions
- Beispiel: Minimum und Maximum eines Vektors

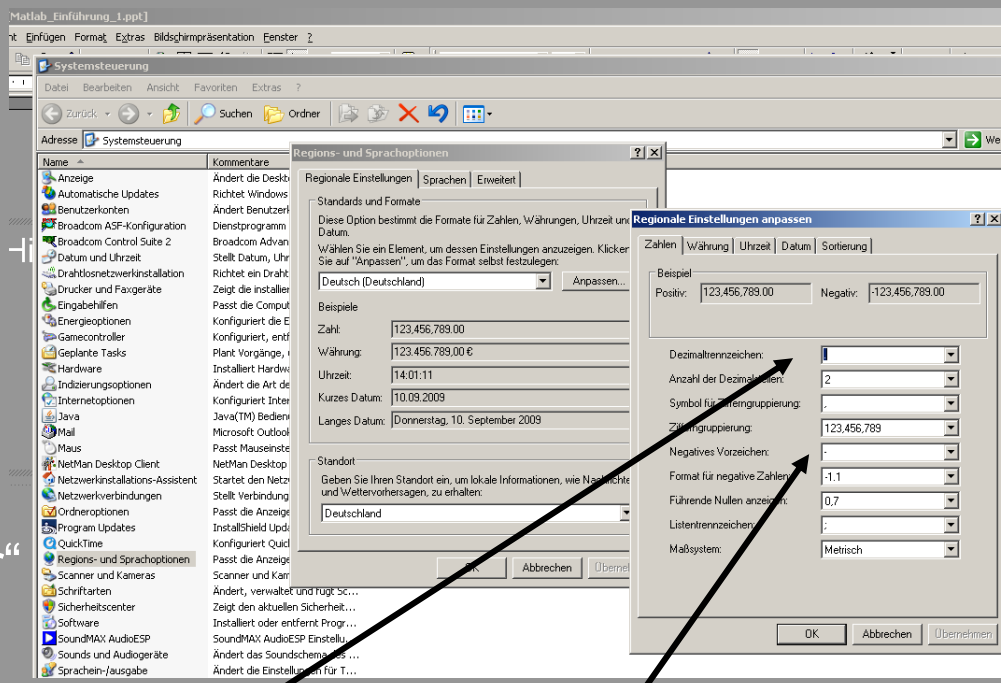


Lesen und Schreiben von Daten

Überblick und Beispiele

Lesen und Schreiben von Daten

Achtung!!



Hinweise:

- Welches Datenformat?
  Ascii = „mit jedem Editor lesbar“
  binär = „nicht mit Editor lesbar“
- Darstellung von Zahlen:
  „Punkt, Punkt, Komma, Strich ...“

- Matlab versteht nur einen „ . “ als Dezimalstelle
- Einstellung in Windows: „Start / Systemsteuerung / Regions- und Sprachoptionen / Regionale Einstellungen“
  „Dezimalzeichen = . “ ; „Symbol für Ziffergruppierung = ,“

Lesen und Schreiben von Daten

Übersicht

Wofür	Befehl	Comment
▪ Speichern ‚workspace‘	save	‚*.mat‘
Laden mat-file	load	Matlab-eigenes Format
▪ Formatierte Ascii-Daten	load / save textread	„nur Zahlen“
▪ Unformatierte Ascii-Daten	fscanf / fprintf	„mit Header“
▪ CSV-Daten	csvread / csvwrite	
▪ Excel-Daten	xlsread / xlswrite	
▪ Binäre Daten	fread / fwrite	
▪ Bildformate	imread / imwrite	für bmp, jpg etc.

...wie überall in Matlab: z. B. ‚help imread‘ eingeben
... und Beispiele kopieren, rauben und anpassen!

Lesen und Schreiben von Daten - mat-files

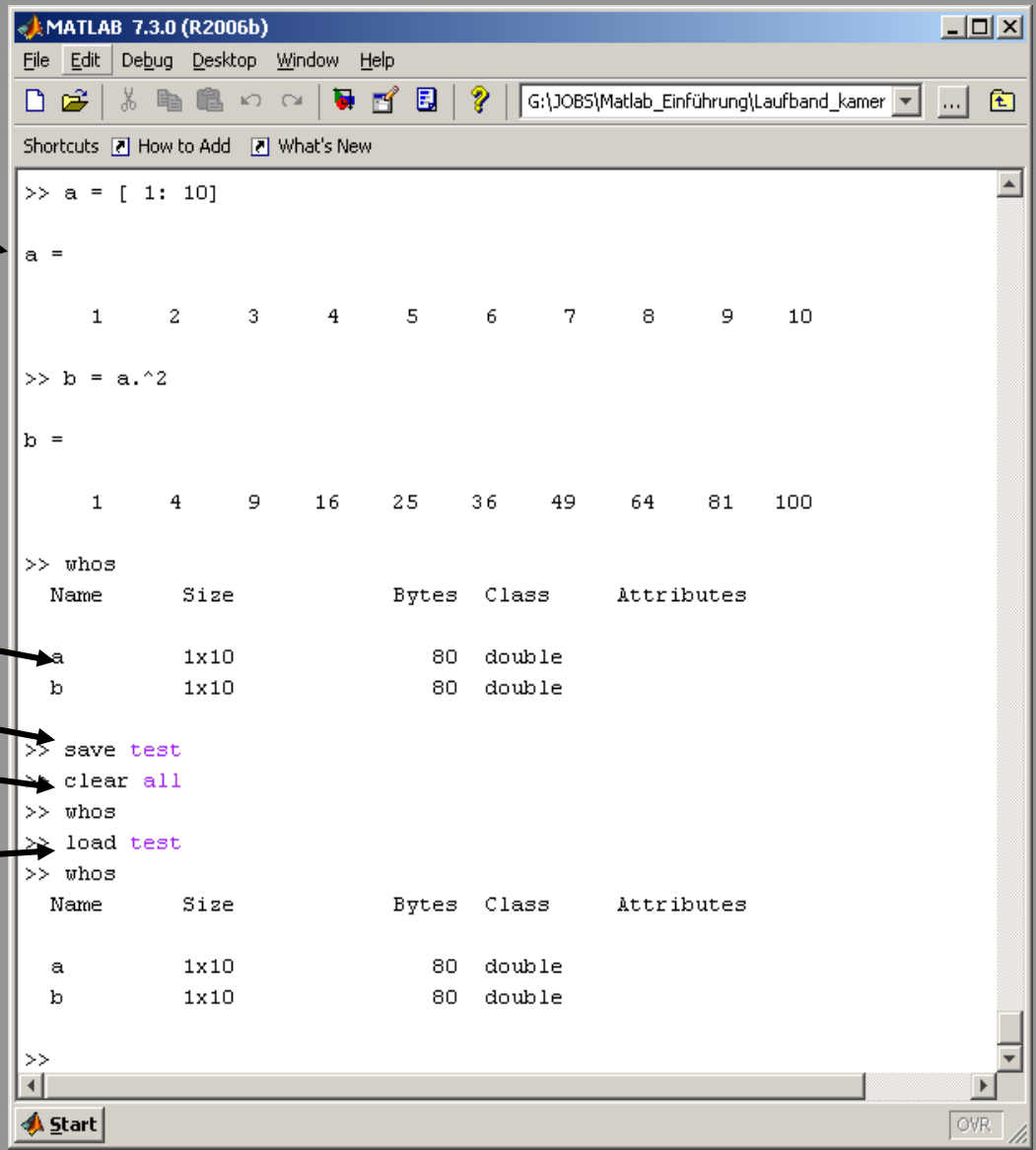
Erzeugen von Daten

Daten im Workspace

Speichern als ‚test.mat‘

Löschen des Workspace

Laden von ‚test.mat‘



Lesen und Schreiben von Daten - Erzeugung von Dateinamen

Ziel: Zusammensetzung von Dateinamen aus Einzelteilen

Zeichenkette: string

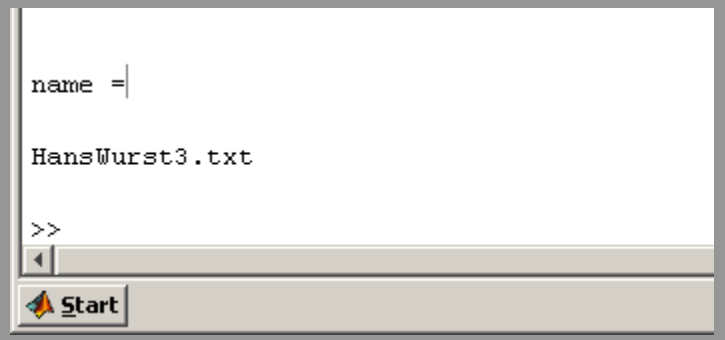
Kein String! sondern ‚double‘

Umwandeln einer Zahl in einen string

Zusammensetzten des gesamten Dateinamens ‚concatenate‘

Resultat im Command Window

```
clear all

name1 = 'Hans'
name2 = 'Wurst'
Zahl = 3
Endung='.txt'

whos

Nummer = num2str(Zahl)

name = cat(2,name1,name2,Nummer,Endung)
```



Aufgabe: Erzeugen von Dateinamen

Aufgabe:
Erzeugen Sie die Dateinamen

‚Albert_Einstein_1.asc'
‚Albert_Einstein_2.asc'
‚Albert_Einstein_3.asc'
....
‚Albert_Einstein_100.asc'

durch Verwendung einer for- oder while- Schleife!

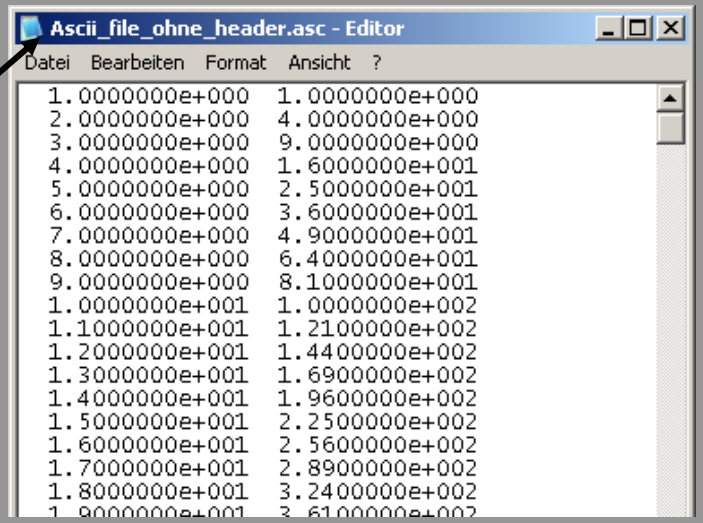
Aufgabe: Erzeugen von Dateinamen - Lösung

```
clear all

name1 = 'Albert'
name2 = '_Einstein_'

Endung='.asc'

for i = 1: 111

    Nummer = num2str(i);
    name = cat(2,name1,name2,Nummer,Endung)

end
```

Lesen und Schreiben von Daten - unformatierte Daten

Daten in File:

Formatiert, d.h. ohne Header / nur Zahlen

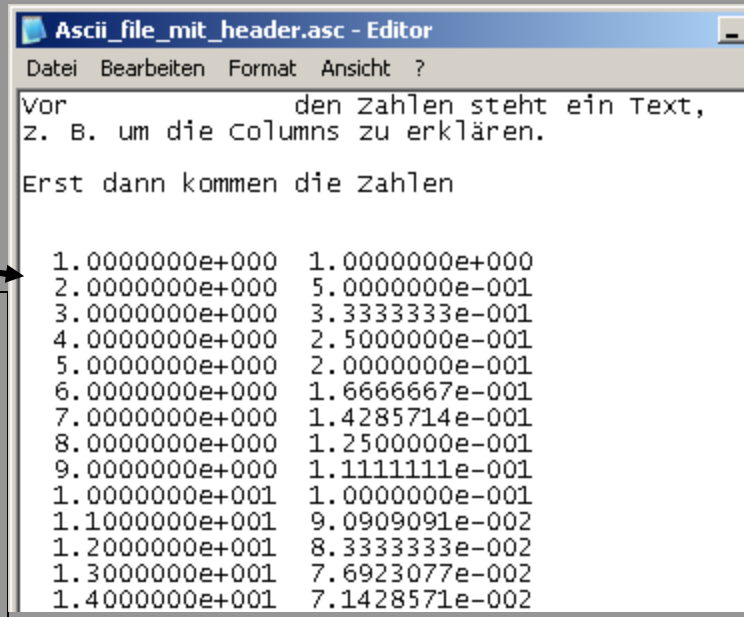



```
clear all

a=load('Ascii_file_ohne_header.asc');

plot(a(:,1),a(:,2))
```

Lesen und Schreiben von Daten - formatierte Daten

Daten in File:

Header, d.h. Text vor Zahlen



File öffnen

18 Wörter lesen

alle Zahlen lesen

Text am Ende lesen

File schließen

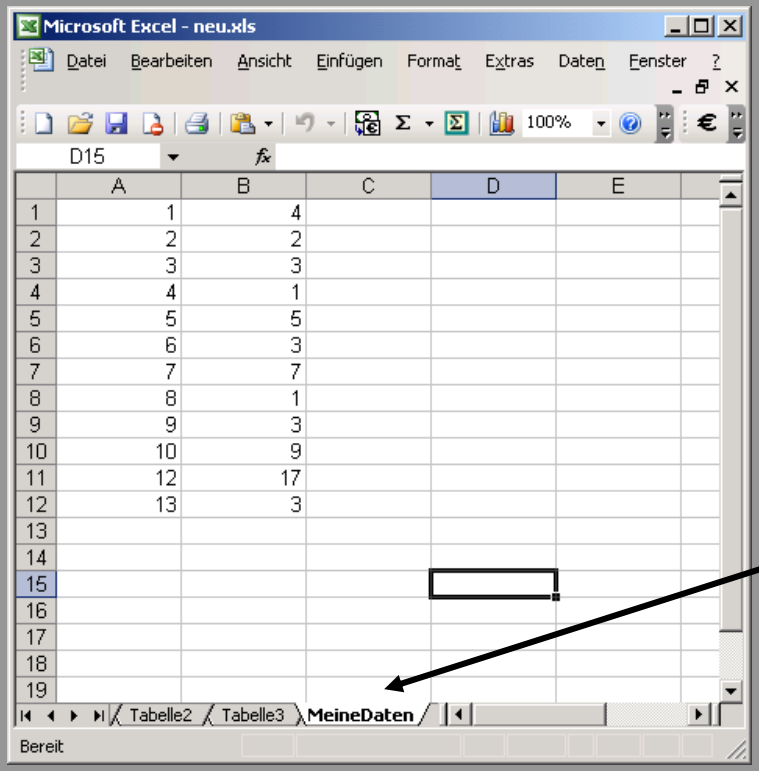
```
fid=fopen('Ascii_file_mit_header.asc');

header=fscanf(fid,'%s',18)

daten=fscanf(fid,'%15e',[2,inf]);
rest=fscanf(fid,'%s',inf)

plot(daten(1,:),daten(2,:))

fclose(fid);
```

Matlab- und MS-Excel (xls)

Lesen von Werten

aus einem Files
in eine Variable

```
data=xlsread('Beispiel_MatExcel_1.xls');

plot(data(:,1),data(:,2))

data(:,3)=data(:,2).^2;

xlswrite('neu.xls',data(:,3));

xlswrite('neu.xls',[data(:,1) data(:,2)],'MeineDaten');
```

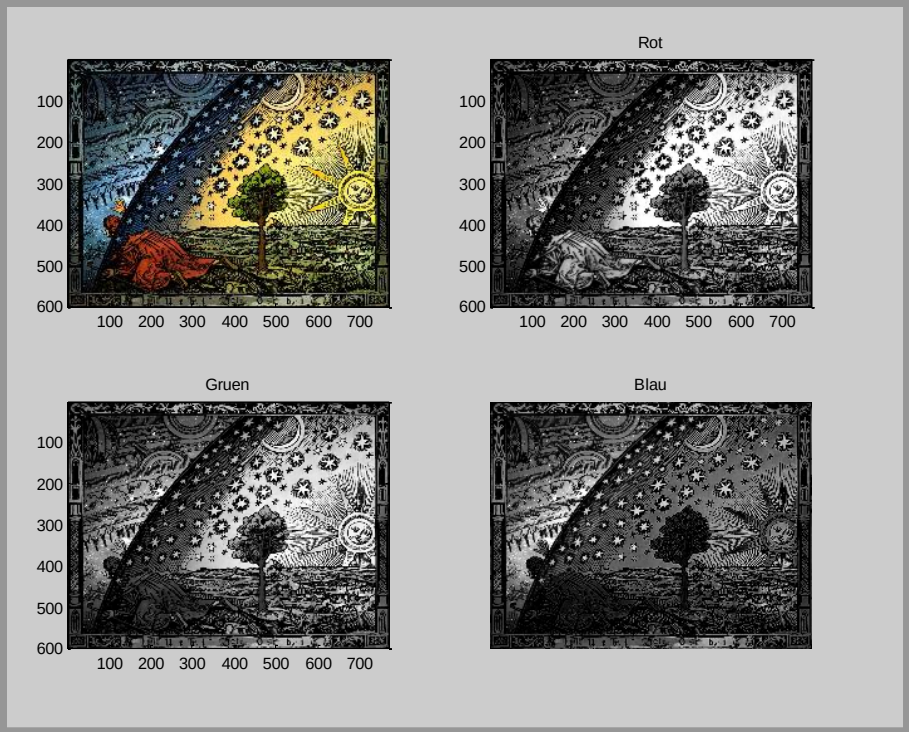


Schreiben von Werten

in ein Excel-File, das
erstellt wird

Lesen / Schreiben von Bildern

- Für Standard – Bildformate
  imread / imwrite




```
clear all
bild = imread('Universum.jpg');

figure(1)
subplot(2,2,1)
imagesc(bild)

rot=bild(:,:,1);
subplot(2,2,2)
imagesc(rot)
title('Rot')
colormap gray

gruen=bild(:,:,2);
subplot(2,2,3)
imagesc(gruen)
title('Gruen')

blau=bild(:,:,3);
subplot(2,2,4)
imagesc(blau)
title('Blau')
axis off
```

3 Farbebenen:
Selektion von ‚rot'

Intensität - Farb-Zuordnung

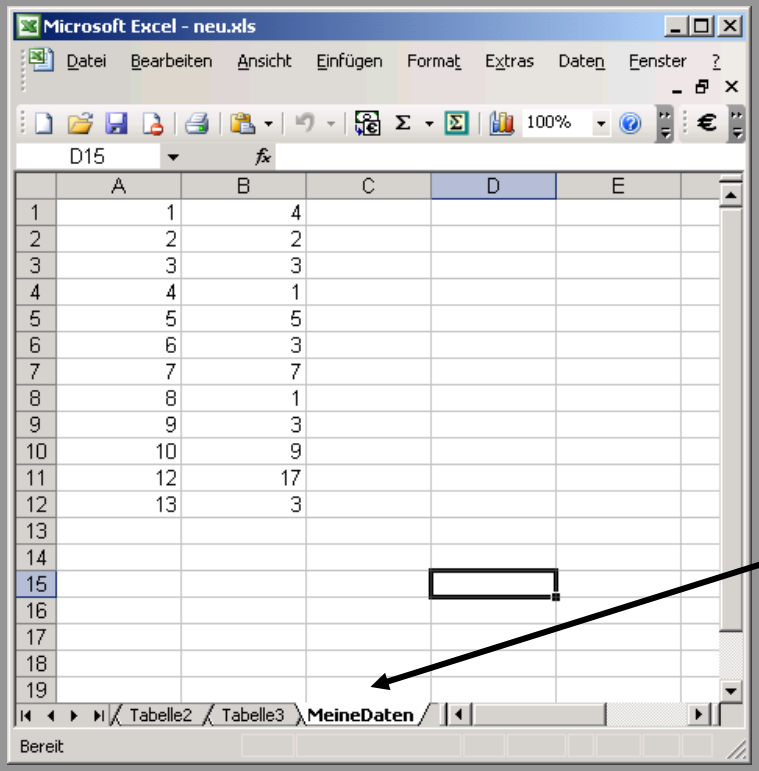
Achsenbeschriftung ‚off'

Matlab- und MS-Excel

Lesen von Werten

aus einem Files
in eine Variable

```
data=xlsread('Beispiel_MatExcel_1.xls');

plot(data(:,1),data(:,2))

data(:,3)=data(:,2).^2;

xlswrite('neu.xls',data(:,3));

xlswrite('neu.xls',[data(:,1) data(:,2)],'MeineDaten');
```



Schreiben von Werten

in ein File

2. Kapitel: Anwendungen und Beispiele aus Medizintechnik und Sportwissenschaft

Aufgabe: EEG / Evozierte Potentiale - Hintergrund

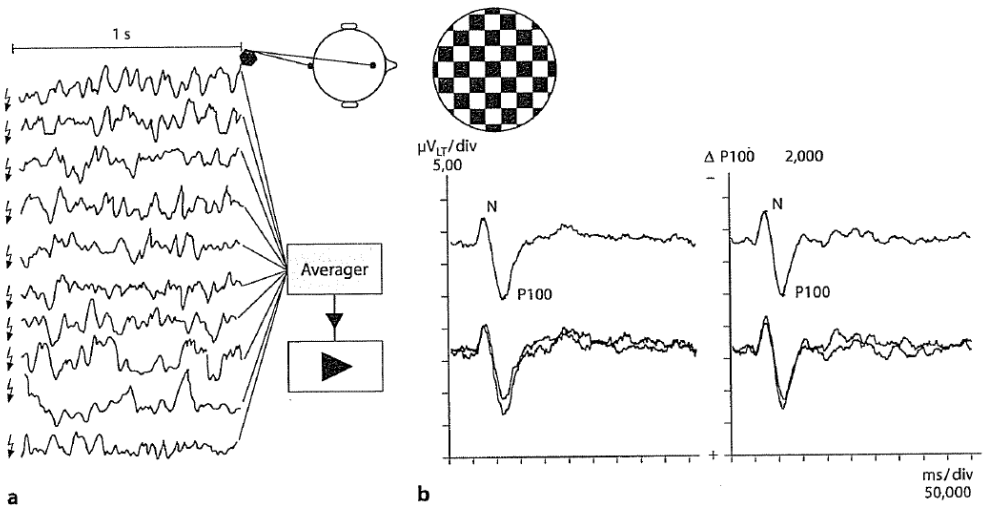


Abb. 2.8. a Registrierung der visuellen Reaktionspotentiale nach Stimulation mit Schachbrettmuster. Durch elektronische Mittelung einer Anzahl einzelner EEG-Abschnitte (*links*) wird die reizabhängige Spannungsänderung im EEG herausgehoben (*rechts*). (Aus Vogel 1981) b Visuell evozierte Potentiale (TV-Stimulation) im Seitenvergleich. *Untere Zeilen* 2 reproduzierte Einzeldurchgänge; *obere Zeilen* summiertes Potential. (Aus Hacke 1986)

Ziel: Auswertung von EEG-Daten während einer visuellen Stimulation (wechselndes Schachbrettmuster auf Bildschirm).

Dabei wurde eine elektrische Spannung mit Elektroden am Kopf aufgenommen und von einem Rechner erfasst. Auf diesen äußeren Reiz (Stimulus) bilden sich durch die neuronale Aktivität Änderungen der Spannung aus, die allerdings sehr klein sind und nicht direkt zu sehen sind, da das Rauschen in den Daten groß ist. Daher muss für eine Bestimmung dieser Potentiale ein Mittelwert über Messungen nach vielen Stimuli gebildet werden, um die elektrische Antwortfunktion zu erkennen.

Aufgabe: EEG / Evozierte Potentiale

In den vorliegenden Daten wurde ein sogenanntes „Visual Evoked Potentials“ (VEP) durch Präsentation eines Schachbrettmusters in einem Probanden erzeugt. Die Antwortkurven von VEPs geben neurologischen Aufschluss über z.B. Epilepsien.

Datenfile: EEG_Evoziertes_Potential.txt enthält die Stimuluszeitpunkte in der ersten Spalte (Stimulusfrequenz: 3 Hz) und das gemessene EEG-Signal in der zweiten Spalte. Die Aufnahmefrequenz war 1000 Hz, d.h. 1 ms liegt zwischen aufeinander folgenden Messpunkten.

Öffnen der Datei mit ‚load’.

Die Stimuli sind durch eine ‚1’ markiert, ansonsten ‚0’.

Aufgabe:

- Die Messwerte sollen bezüglich des Stimulus gemittelt (englisch: average) werden.

- Die Rohdaten Messwerte sollen zusammen mit dem Mittelwert in einer Graphik dargestellt werden.

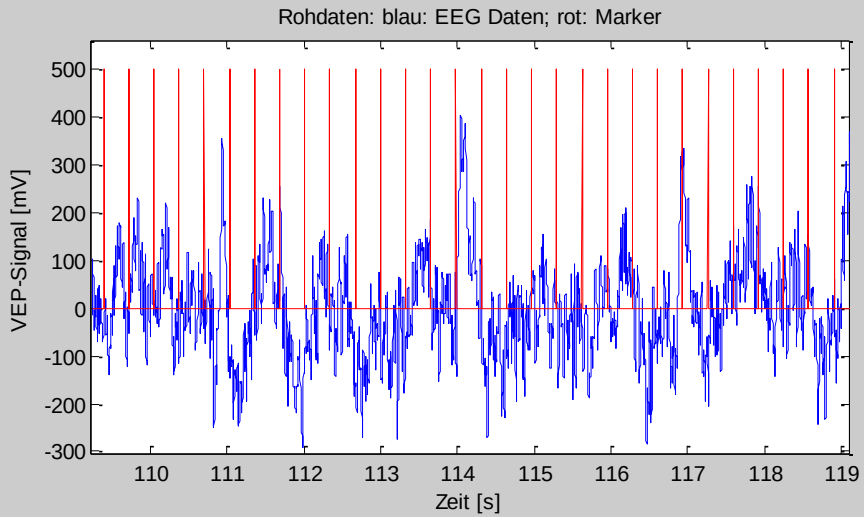
Hinweis: der Befehl ‚find’ kann zum Bestimmen der Stimuli verwendet werden. Alternativ kann eine for-Schleife mit einer if-Abfrage verwendet werden.

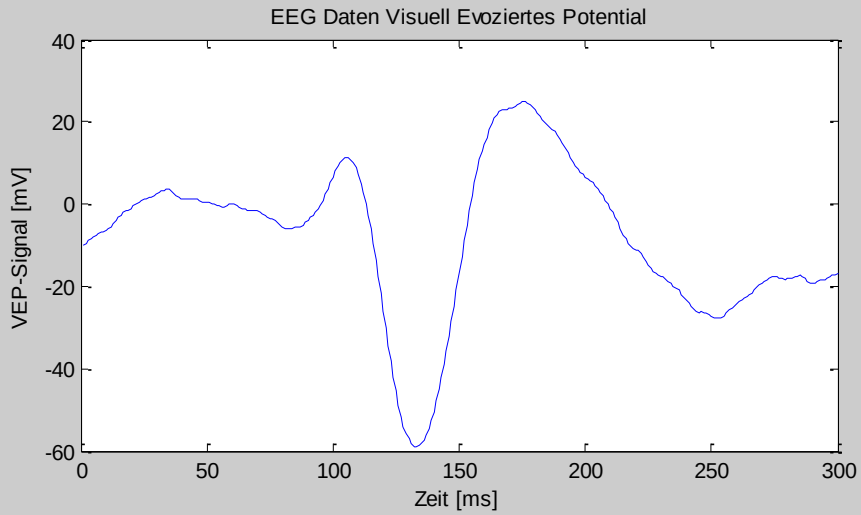
Aufgabe: EEG / Evozierte Potentiale - Lösung

```
% EEG_Evoziertes_Potential.m
clear all
 eeg_data=load('EEG_Evoziertes_Potential.txt');
laenge=300;
avg_data=zeros(laenge,1);
trigger=find(eeg_data(:,1)==1);

for i=1:size(trigger)-1
      avg_data=eeg_data(trigger(i):trigger(i)+laenge-1,2)+avg_data;
end
avg_data=avg_data/i;
% Zeitachse: 1000 Hz
time=[1:size(eeg_data,1)]/1000;
figure(1)
subplot(2,1,1)
plot(time,eeg_data(:,2))
hold on
plot(time,eeg_data(:,1)*500,'r')
xlabel('Zeit [s]')
ylabel('VEP-Signal [mV]')
title('Rohdaten: blau: EEG Daten; rot: Marker')
```

```
subplot(2,1,2)
plot(avg_data)
xlabel('Zeit [ms]')
ylabel('VEP-Signal [mV]')
title('EEG Daten Visuell Evoziertes Potential')
```

Aufgabe: EEG / Evozierte Potentiale - Lösung





Ergebnis:

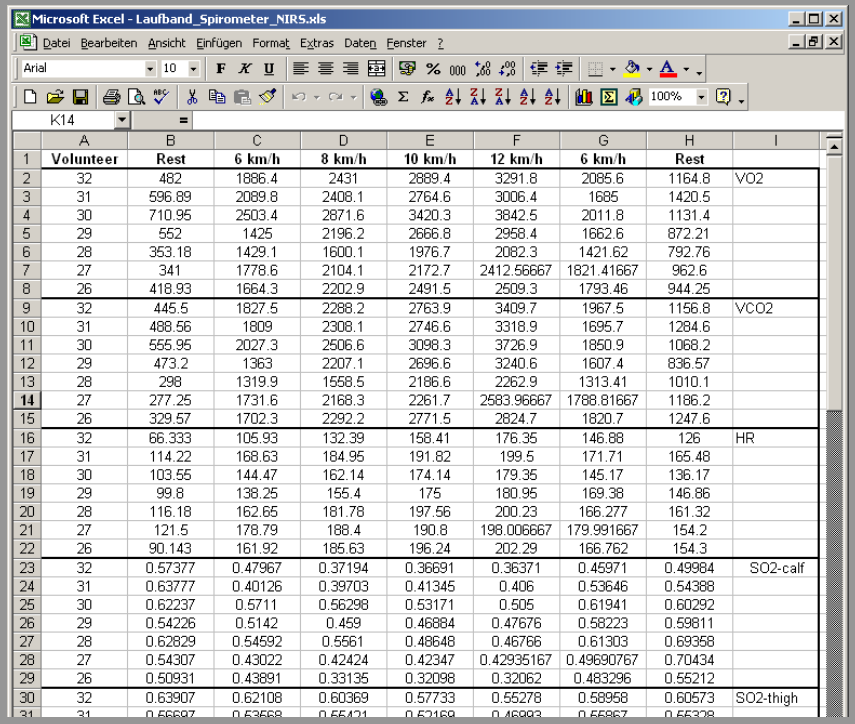
Zeitverlauf der Spannung nach dem Stimulus (Zeit = 0 ms)

Minimum bei ca. 130 ms

Aufgabe: Spirometrische Daten

In der Excel-Datei
„Laufband_Spirometer_NIRS.xls“
liegen Messungen von den Probanden (Nr. 26 – 32) während einer Laufstudie vor. Gemessen wurden die spirometrischen Größen VO2 und VCO2, die Herzrate (HR).
Lesen Sie die Daten in Matlab ein und stellen Sie sie graphisch als Funktion der Geschwindigkeit dar.

Berechnen Sie die Mittelwerte (‚mean’) über die Probanden und plotten Sie diese Werte gesondert, wobei die Standardabweichung (‚std’) die Fehlerbalken sein sollen (Befehl ‚errorbar’)!
Achten Sie bei ‚mean’ und ‚std’ darauf, ob die Operation auf Zeilen oder Spalten bezogen wird (Hinweise: Transponieren der Daten, ‚transpose’, oder unter ‚help mean’.).


Microsoft Excel - Laufband_Spirometer_NIRS.xls

Datei Bearbeiten Ansicht Einfügen Format Extras Daten Fenster ?

K14 =

	A	B	C	D	E	F	G	H	I
1	Volunteer	Rest	6 km/h	8 km/h	10 km/h	12 km/h	6 km/h	Rest	
2	32	482	1886.4	2431	2889.4	3291.8	2085.6	1164.8	VO2
3	31	596.89	2089.8	2408.1	2764.6	3006.4	1685	1420.5	
4	30	710.95	2503.4	2871.6	3420.3	3842.5	2011.8	1131.4	
5	29	552	1425	2196.2	2666.8	2958.4	1662.6	872.21	
6	28	353.18	1429.1	1600.1	1976.7	2082.3	1421.62	792.76	
7	27	341	1778.6	2104.1	2172.7	2412.56667	1821.41667	962.6	
8	26	418.93	1664.3	2202.9	2491.5	2509.3	1793.46	944.25	
9	32	445.5	1827.5	2288.2	2763.9	3409.7	1967.5	1156.8	VCO2
10	31	488.56	1809	2308.1	2746.6	3318.9	1695.7	1284.6	
11	30	555.95	2027.3	2506.6	3098.3	3726.9	1850.9	1068.2	
12	29	473.2	1363	2207.1	2696.6	3240.6	1607.4	836.57	
13	28	298	1319.9	1558.5	2186.6	2262.9	1313.41	1010.1	
14	27	277.25	1731.6	2168.3	2261.7	2583.96667	1788.81667	1186.2	
15	26	329.57	1702.3	2292.2	2771.5	2824.7	1820.7	1247.6	
16	32	66.333	105.93	132.39	158.41	176.35	146.88	126	HR
17	31	114.22	168.63	184.95	191.82	199.5	171.71	165.48	
18	30	103.55	144.47	162.14	174.14	179.35	145.17	136.17	
19	29	99.8	138.25	155.4	175	180.95	169.38	146.86	
20	28	116.18	162.65	181.78	197.56	200.23	166.277	161.32	
21	27	121.5	178.79	188.4	190.8	198.006667	179.991667	154.2	
22	26	90.143	161.92	185.63	196.24	202.29	166.762	154.3	
23	32	0.57377	0.47967	0.37194	0.36691	0.36371	0.45971	0.49984	SO2-calf
24	31	0.63777	0.40126	0.39703	0.41345	0.406	0.53646	0.54388	
25	30	0.62237	0.5711	0.56298	0.53171	0.505	0.61941	0.60292	
26	29	0.54226	0.5142	0.459	0.46884	0.47676	0.58223	0.59811	
27	28	0.62829	0.54592	0.5561	0.48648	0.46766	0.61303	0.69358	
28	27	0.54307	0.43022	0.42424	0.42347	0.42935167	0.49690767	0.70434	
29	26	0.50931	0.43891	0.33135	0.32098	0.32062	0.483296	0.55212	
30	32	0.63907	0.62108	0.60369	0.57733	0.55278	0.58958	0.60573	SO2-thigh
31	31	0.56697	0.53568	0.55421	0.52169	0.46993	0.55867	0.56328	

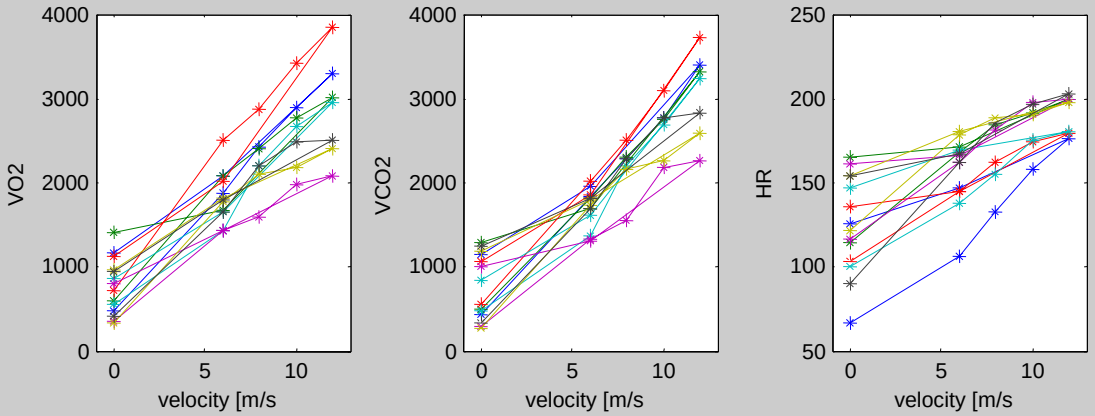
Aufgabe: Spirometrische Daten - Lösung

```
clear all
clf
data = xlsread('Laufband_Spirometer_NIRS.xls');

% Geschwindingkeit [m/s]
Vel = [0 6 8 10 12 6 0 ]

VO2 = data(1:1+6,2:end);
VCO2= data(8:8+6,2:end);
HR  = data(15:15+6,2:end);

VO2=VO2';
VCO2 = VCO2';
HR = HR';

%Mittelwerte und Std
m_VO2=mean(VO2');
s_VO2=std(VO2');
m_VCO2=mean(VCO2');
s_VCO2=std(VCO2');
m_HR=mean(HR');
s_HR=std(HR');
```

```
figure(1)
subplot(2,3,1)
plot(Vel,VO2,'-*')
xlabel('velocity [m/s')
ylabel('VO2')
xlim([-1 13])

subplot(2,3,2)
plot(Vel,VCO2,'-*')
xlabel('velocity [m/s]')
ylabel('VCO2')
xlim([-1 13])

subplot(2,3,3)
plot(Vel,HR,'-*')
xlabel('velocity [m/s]')
ylabel('HR')
xlim([-1 13])
```

```
% #### Mean + Std
subplot(2,3,4)
errorbar(Vel,m_VO2,s_VO2,'-*')
ylabel('VO2')
xlim([-1 13])

subplot(2,3,5)
errorbar(Vel,m_VCO2,s_VCO2,'-*')
xlabel('velocity [m/s]')
ylabel('VCO2')
xlim([-1 13])

subplot(2,3,6)
errorbar(Vel,m_HR,s_HR,'-*')
ylabel('HR')
xlim([-1 13])
```

Aufgabe: Spirometrische Daten - Lösung



Aufgabe: Markertracking

Das Ziel ist es, mit einfachsten Mitteln ein Video-basiertes Markertracking zu erstellen. In den Bildern ‚bild604_1.jpg' bis ‚bild604_17.jpg' wird ein Proband während des Laufens beobachtet.

- Lesen Sie alle Bilder nacheinander ein.
- Selektieren Sie die blaue Farbebene jedes Bildes.
- Setzen Sie alle Bildpunkte, deren Wert kleiner als ein Schwellwert ist (z. B. 140) als ‚0', ansonsten ‚1'. Dazu können zwei for-loops und eine if-Abfrage verwendet werden.
- Lassen Sie die Bilder nacheinander als Film anzeigen (‚imagesc'), sowohl Roh-Bild als auch Marker-Bild mit den ‚0' oder ‚1'.
- Es sollte dann nur ein Marker am Fuß übrig bleiben, deren Pixelposition berechnet werden kann (z.B. über ‚mean').
- Stellen Sie diese Pixelposition für x- und y-Richtung als Funktion der Bildnummer graphisch dar.
- Stellen Sie die Pixelposition von y-Richtung gegen y-Richtung graphisch dar.

(Niemand würde eine Markertracking wirklich so programmieren, da es zu langsam läuft. Aber die notwendigen Programmierschritte werden deutlich.)

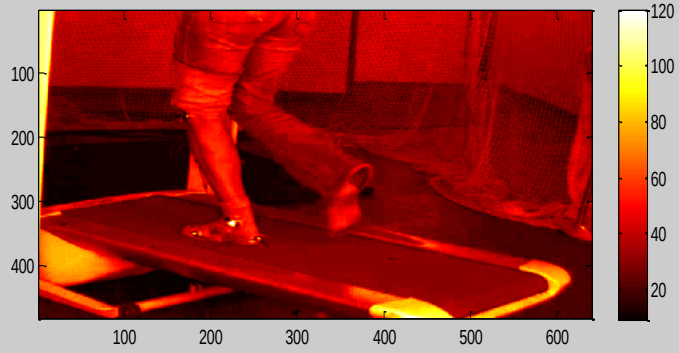
Aufgabe: Markertracking - Lösung

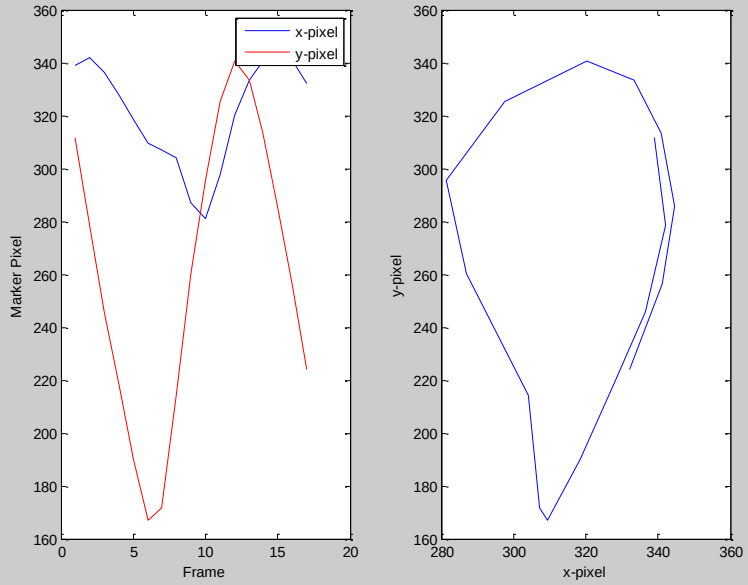
```
clear all
name1 = 'bild'
name2 = '604_'
Endung='.jpg'

for n = 1: 17
   Nummer = num2str(n);
   name =    cat(2,name1,name2,Nummer,Endung)
   a = imread(name);
   b= a(:,:,3);

   c=zeros(size(b));
   for i = 1: size(b,1)
      for j = 1:size(b,2)
         if b(i,j) > 140
            c(i,j)=1;
         end
      end
   end
```

```
   figure(1)
   subplot(2,1,1)
   imagesc(b,[9 120])
   colorbar
   colormap hot

   subplot(2,1,2)
   imagesc(c)
   colorbar

   [row, col]=find(c);

   x(n)=mean(row);
   y(n)=mean(col);

end
```

```
figure(3)
subplot(1,2,1)
plot(x)
hold on
plot(y,'r')
hold off
xlabel('Frame')
ylabel('Marker Pixel')
legend('x-pixel','y-pixel')

subplot(1,2,2)
plot(x,y)
xlabel('x-pixel')
ylabel('y-pixel')
```

Aufgabe: Markertracking - Lösung





Aufgabe: Muskeloxygenierung

Im Datenfile NIRS_data.txt sind Messungen der Sauerstoffsättigung des Hämoglobin im Oberschenkelmuskel – gemessen mit der Nahinfrarotspektroskopie - während eines Stufenprotokols auf dem Ergometer.

1. Spalte: Zeit in min
2. Spalte: Sauerstoffsättigung (SO2) in %.
3. Marker: Wenn ‚1' vorliegt, wurde die Leistung auf dem Ergometer um 30 W erhöht.

Aufgabe:

- Stellen Sie die Daten als Funktion der Zeit dar.
- Schreiben Sie ein Programm zum Glätten der Daten.
- Mitteln Sie die Daten bezogen auf jede der Leistungsstufen und stellen Sie die Abnahme der Sauerstoffsättigung als Funktion der Leistung dar.

Aufgabe: Muskeloxygenierung - Lösung

```
clear all

data = load('NIRS_data.txt');

figure(1)
subplot(2,1,1)
plot(data(:,1),data(:,2))
hold on
plot(data(:,1),data(:,3),'r')
hold off
xlabel('Zeit [min]')
ylabel('SO2 [%]')

Marker = find(data(:,3));

for i = 1:length(Marker)-1
   Mittelwert_SO2(i)=mean(data(Marker(i):Marker(i+1),2));
end

Power= ([1:10]-1)*30+ 40;

subplot(2,1,2)
plot(Power,Mittelwert_SO2,'*-r')
xlabel('Leistung [W]')
ylabel('SO2 [%]')
```