Einführung in das Programmieren mit Matlab, Teil 3/4

Thomas Dunst

Mathematisches Institut
Universität Tübingen

(e–mail: progtutor@na.uni-tuebingen.de)

11. Oktober 2012

Wiederholung:

Was haben wir bislang gemacht:

- Kontrollanweisungen (Schleifen, Verzweigungen)
- Vektoren und Matrizen (Initialisierung, Operationen)

Sind stehengeblieben bei:

- Funktionen und Skripte (Aufbau, Funktionsweise)

```
 1  A = [1 2 3; 4 5 6; 7 8 9; 10 11 12];
 2  [n m] = size(A);
 3  i = 1;   % Diese Zeile soll gestrichen werden
 4  if i==1 % Fall: erste Zeile streichen
 5      B = A(2:n,:);
 6  elseif i==n % Fall: letzte Zeile streichen
 7      B = A(1:n-1,:);
 8  else % Sonst
 9      B = A(1:i-1,:);
10      B(i:n-1,:) = A(i+1:n,:);
11  end
12  B % Ausgeben
```

Gliederung

Lösungen der Aufgaben des 2. Übungsblattes

Funktionen und Skripte
- Definition einer Funktion
- Auslagern von Code-Blöcken in Funktion
- Beispiel: Funktion
- wichtige Funktionen in Matlab

Visualisierung
- Plotten in Matlab: 2D Plot
- Vorgehen beim Plotten
- Beispiel: Plotten
- Bemerkungen zum Plotten
- Plotten in Matlab: 3D Plot

Lösungen der Aufgaben des 2. Übungsblattes:

Aufgabe 1:

```
1  i
2  j
3  n
```

Ausgabe:

```
>> i
ans =    0 + 1.0000i

>> j
ans =    0 + 1.0000i

>> n

??? Undefined function or variable 'n'.
```

$\Rightarrow$ i und j sind in Matlab vordefiniert als komplexe Zahl i, n ist nicht vordefiniert.
Achtung: Vergisst man bei Verwendung von i bzw. j z.B. als Schleifenvariablen die Initialisierung, kann das zu Problemen führen, da MATLAB nicht meldet, das die Variable unbekannt ist.

Lösungen der Aufgaben des 2. Übungsblattes:

Aufgabe 3:
Ausgabe:

```
ii =  0.75000
ii =  1.2500
ii =  1.7500
ii =  2.2500
ii =  2.7500
ii =  3.2500
ii =  3.7500
ii =  4.2500
ii =  4.7500
```

$\Rightarrow$ Die for-Schleife wird 9 mal durchlaufen und dass Inkrement ist offensichtlich 0.5. Ausgegeben wird das aktuelle `ii` -0.25.
Die Änderung von `ii` in der Schleife hat nur Wirkung auf die restlichen Zeilen im Schleifeninneren.
Die for-Schleife hingegen setzt für jeden neuen Durchlauf das `ii` um 0.5 hoch, und zwar ausgehend von 1 und nicht vom aktuellen Wert `ii` im Schleifeninneren.

Lösungen der Aufgaben des 2. Übungsblattes:

Die while-Schleife:

```
1  ii = 1
2  while ii <= 5
3    ii = ii - 0.25
4  end
```

erzeugt eine Endlosschleife, da `ii` beginnend mit einem Wert kleiner als 5 monoton fällt in Schrittweiten von 0.25 . Das Abbruchkriterium wird nie erfüllt.
$\Rightarrow$ Endlosschleifen kann man in Matlab mit der Tastenkombination `Strg-C` bzw. `Ctrl-C` abbrechen.

Ausgabe:

```
ii =  0.75000
ii =  0.50000
ii =  0.25000
ii =  0
ii = -0.25000
ii = -0.50000
ii = -0.75000
ii = -1
ii = -1.2500
ii = -1.5000
ii = -1.7500
ii = -2
ii = -2.2500
:
```

Lösungen der Aufgaben des 2. Übungsblattes:

Aufgabe 4 und 5:

```
 1  a = [ 2 2.5 3 3.5]'     % Spaltenvektor 4 x 1
 2  b = [ 1 0 -1 -2 ]'      % Spaltenvektor 4 x 1
 3  n = length(a);          % Laenge des Vektors ermitte
 4  c = zeros(4,1);         % Vektor c initialisieren
 5  for i = 1 : n
 6    c(i) = b(i) - a(i)    % Standardmaessig erzeugt
 7                          % Matlab Zeilenvektoren
 8                          % durch Vorinitialisierung
 9                          % ist c Spaltenvektor
10  end
11  c                       % c ausgeben
12  d = b - a               % zur Kontrolle, Matlab
13                          % erzeugt Spaltenvektoren
```

Ausgabe: (umgebrochen in 2 Spalten)

```
c =  -1.0000   0.0000   0.0000   0.0000        d =
c =  -1.0000  -2.5000   0.0000   0.0000         -1.0000
c =  -1.0000  -2.5000  -4.0000   0.0000         -2.5000
c =  -1.0000  -2.5000  -4.0000  -5.5000         -4.0000
c =  -1.0000  -2.5000  -4.0000  -5.5000         -5.5000
```

Lösungen der Aufgaben des 2. Übungsblattes:

Aufgabe 6:

Seien $v, w \in \mathbb{R}^n$

- Skalarprodukt: $v^T w$
- Dyadisches Produkt: vw^T
- Punktweise Multiplikation: $v(i) \cdot w(i)$ für alle $i = 1, \ldots, n$

In MATLAB :

```
1  v = [1; 2];
2  w = [3; 4];
3  v' * w % Skalarprodukt
4  v * w' % Dyadisches Produkt
5  v .* w % Punktweise Multipl.
```

Ausgabe:

```
ans =
    11
ans =
     3     4
     6     8
ans =
     3
     8
```

Lösungen der Aufgaben des 2. Übungsblattes:

Aufgabe 6:

```
 1  v = [ 1; 2];
 2  w = [ 3; 4];
 3  % Skalarprodukt
 4  r = 0;
 5  n = length(v);
 6  for i = 1 : n
 7    r = r + v(i) * w(i)
 8  end
 9
10  r = [] % r loeschen
11  % dyadisches Produkt
12  % i-te Zeile von r
13  for i = 1 : n
14    % j-te Spalte von r
15    for j = 1 : n
16      r(i,j) = v(i) * w(j)
17    end
18  end
```

Ausgabe:

```
r = 3
r = 11

r = []
r = 3
r =
     3   4
r =
     3   4
     6   0
r =
     3   4
     6   8
```

Lösungen der Aufgaben des 2. Übungsblattes:

Aufgabe 7a): Arithmetisches Mittel berechnen

```
 1  % Verwende Testvektor:
 2  v = [-4   2 -5 -7 -1 6 -3];
 3
 4  s = 0; % Initialisierung Summe
 5  n = length(v); % Bestimmen Vektordim.
 6  for i = 1 : n
 7     s = s + v(i)
 8  end
 9
10  s = s / n
```

Lösungen der Aufgaben des 2. Übungsblattes:

Aufgabe 7b): Euklidische Norm berechnen

```
 1  s = 0 % Initialisierung Summe
 2
 3  n = length(v); % Bestimmen Vektordim.
 4  for i = 1 : n
 5    s = s + v(i) * v(i)
 6  end
 7
 8  sqrt(s)
 9
10  % nur zum Test:
11  % Norm mit Matlab-Funktion ausrechnen
12  norm(v, 2)
```

Lösungen der Aufgaben des 2. Übungsblattes:

Aufgabe 7c): Maximums-Norm berechnen

```
 1  s = abs(v(1)) % aktuell groesster Wert
 2
 3  n = length(v); % Bestimmen Vektordim.
 4  for i = 2 : n
 5    % wenn neues v(i) groesser...
 6    if s < abs(v(i))
 7      s = abs(v(i)) % ... uebernehmen
 8    end
 9  end
10  s
11
12  % nur zum Test:
13  % Norm mit Matlab-Funktion ausrechnen
14  norm(v, Inf)
```

Lösungen der Aufgaben des 2. Übungsblattes:

Aufgabe 7d): Betragssummen-Norm berechnen

```
1  s = 0 % Initialisierung Summe
2
3  n = length(v); % Bestimmen Vektordim.
4  for i = 1 : n
5      s = s + abs(v(i))
6  end
7  s
8
9  norm(v, 1)
```

Lösungen der Aufgaben des 2. Übungsblattes:

Aufgabe 8:

```
 1  % Matrix A sei gegeben
 2
 3  % c: Schrittweite fuer Spalte
 4  c = 1   % alle Spalten besuchen
 5  c = 2   % nur alle ungeraden Spalten
 6
 7  [n, m] = size(A)
 8  for i = 1 : n % i-te Zeile
 9    for j = 1 : c : m % i-te Spalte
10      A(i,j)
11    end
12  end
```

Lösungen der Aufgaben des 2. Übungsblattes:

Aufgabe 9a):

```
 1  % Matrix A sei gegeben
 2  % Vektor v sei gegeben
 3  % Es soll Av berechnet werden
 4
 5  [n,m] = size(A)
 6
 7  erg = zeros(n,1); % Initialisieren der Summe
 8  for ii=1:n
 9    for jj=1:m
10      erg(ii) = erg(ii) + A(ii,jj)*v(jj);
11    end
12  end
```

Lösungen der Aufgaben des 2. Übungsblattes:

Aufgabe 9b):

```
 1  % Matrix A sei gegeben
 2  % Vektor v sei gegeben
 3  % Es soll v'A berechnet werden
 4
 5  [n,m] = size(A)
 6
 7  erg = zeros(1,m); % Initialisieren der Summe
 8  for ii=1:m
 9    for jj=1:n
10      erg(ii) = erg(ii) +  v(jj)*A(ii,jj);
11    end
12  end
```

Lösungen der Aufgaben des 2. Übungsblattes:

Aufgabe 11:

```
1  function tfahr = celinfahr(tcel)
2  % Umrechnung von Celsius in Fahrenheit
3  % Input: tcel (Temp. in Celsius)
4  % Output: tfahr (Temp. in Fahrh.)
5      tfahr = tcel * (9/5) +32;
6  end
```

Aufruf mit:

```
1  celinfahr(30) % Wenn man 30 Grad Cel
2                % umrechnen moechte
```

Funktionen und Skripte

MATLAB unterscheidet zwischen Skripten und Funktionen:

- Skripte sind Textdateien, die Anweisungen enthalten, die man genauso gut im Command Window einzeln eintippen könnte.
- Funktionen sind auch Textdateien mit folgenden Unterschieden zu Skriptdateien:
  - Funktionendateien enthalten spezielle Anweisungen: In der ersten Zeile steht als erstes das Schlüsselwort `function` und weiter hinten in der Zeile der zum Dateinamen gleichlautende Funktionenname.
  - Funktionen können so angelegt werden, dass sie beim Aufruf ein oder mehrere Funktionsargumente erwarten, z.B. `f(x)`
  - Funktionen können so angelegt werden, dass sie nach ihrer Beendigung Rückgabewerte an den Aufrufer liefern.
  - Es besteht die Möglichkeit, über spezielle Kommentare eine Hilfe für die Funktion in Matlab zu importieren.

Definition einer Funktion

Eine eigene Funktion wird in MATLAB in folgender Weise definiert.
Bsp:

```
 1  function [X, Y] = funktionsname(A,B,C)
 2  % Ein paar Worte was die Funktion macht
 3  %
 4  % Input:
 5  % Matrix A: Bedeutung
 6  % Matrix B: Bedeutung
 7  % Matrix C: Bedeutung
 8  % Output:
 9  % Matrix X: Bedeutung
10  % Matrix Y: Bedeutung
11
12      :
13      :
14  end
```

Definition einer Funktion II

- Der Dateiname der im obigen Bsp. definierten Funktion namens `funktionsname` lautet `funktionsname.m`
- Die Funktion liefert als Rückgabewert in einem Feld 2 Variablen passenden Typs zurück: `X, Y`.
- Die Funktion erwartet beim Aufruf 3 Variablen passenden Typs: `A, B, C`.
- Die zurückgegebenen Variablen enthalten die Werte, die diese Variablen am Ende der Ausführung der Funktion hatten. (D.h. die Rückgabewerte müssen nicht speziell irgendwo gesetzt werden oder speziell zurückgegeben werden.)
- Erwartet die Funktion keine Argumente, bleibt die runde Klammer hiner dem Funktionsnamen leer `()`.
- Gibt die Funktion keine Argumente zurück bleibt die eckige Klammer hinter `function` leer `[]`.

Auslagern von Code-Blöcken in eine Funktion

Das Vorgehen:

- Anlegen einer Daten mit dem Namen der zu erzeugenden Funktion (plus `.m` als Endung).
- Erzeugen eines Funktionsgerüsts wie im Beispiel.
- Kopieren des Code-Blocks in die erzeugte Datei.
- Anpassen des Codes, so dass er evtl. übergebene Argumente verarbeitet.
- Anpassen des Codes, so dass er evtl. geforderte Rückgabewerte zurückgibt.
- Testen...
- Achtung: Die übergebenen Datentypen müssen zusammenpassen. Erwartet die Funktion z.B. einen Skalar, erhält aber eine Matrix, beschwert sich Matlab erst, wenn etwa eine Matrixmultiplikation wegen falscher Dimensionen der Beteiligten fehlschlägt.

Beispiel Funktion

Beispiel-Funktion in Datei bsp02.m

```
 1  function [a, b] = bsp02(X, Y)
 2  % Hilfetext...
 3  %
 4  % Input:
 5  % Matrix X, Y   ...
 6  % Output:
 7  % Matrix a, b   ...
 8
 9  a =   X * Y   % Rueckgabevariable a belegen
10  b =   X + Y   % Rueckgabevariable b belegen
11  a = []        % a doch leeren.
12
13  % b enthaelt X + Y
14  % a ist leerer Vektor
15  end
```

Beispiel-Skript

```
1  g1 = 3;
2  [c,d] = bsp02(g1, 2);
3  c  % sollte leer sein
4  d  % sollte 5 ( = 2 + 3) sein.
```

wichtige Funktionen in Matlab

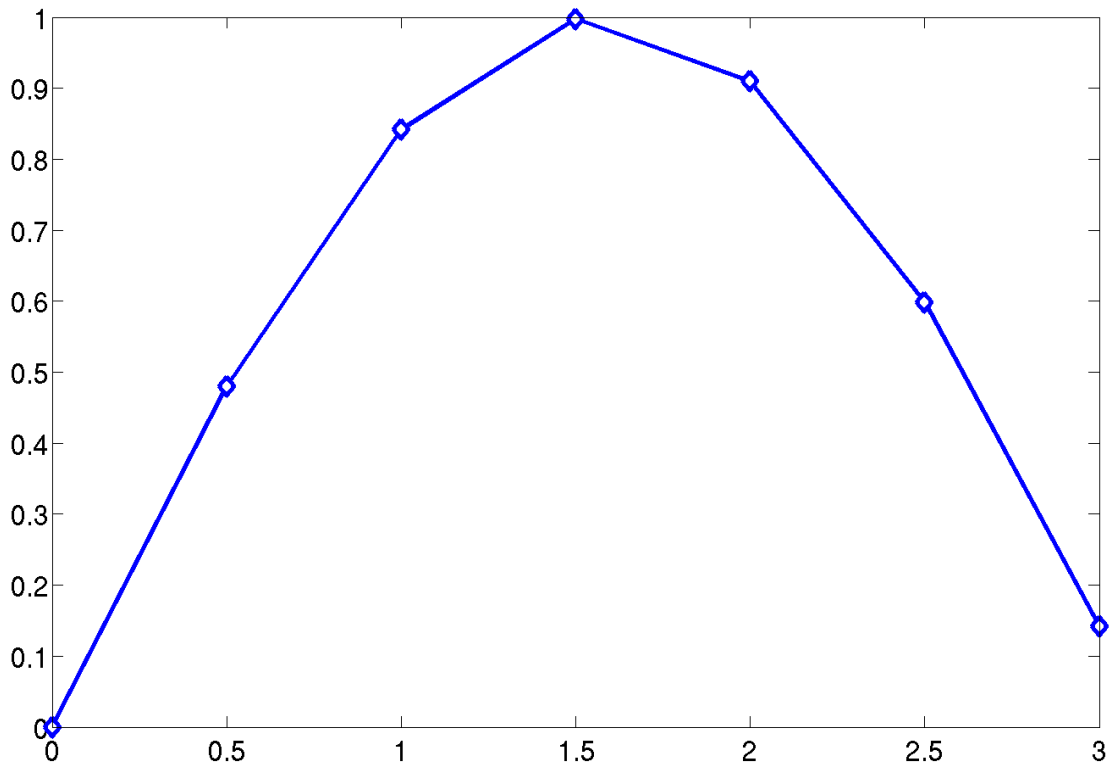
Funktionen, die man in Matlab oft benutzt.

- Trigonometrische Funktionen `sin, cos, tan`
- andere mathematische Funktionen `exp, log, abs, sqrt`
  (Exponentialfunktion, Logarithmusf., Absolutbetragsf., Wurzelf.)
- Verwaltungsfunktionen `length, size`
  Größe von Daten/Variablen bestimmen.
- Ein/Ausgabe-Funktionen `disp, load, save`
  einfache Bildschirmausgabe, laden / speichern von Variablen.
- andere Ein/Ausgabe-Funktionen `fprintf, sprintf`
  mächtigere Ausgabe auf Bildschirm oder in Datei, Zeichenketten erzeugen. (wird nicht besprochen)
- grafische Funktionen `plot`
  Funktionen plotten. (3. Teil der Veranstaltung)
- mächtige Funktionen `lu, qr, norm, max`
  Marix-Zerlegungen, Norm ber., max. (wird nicht besprochen)

Plotten in Matlab: 2D Plot

- 2D-Plot:
  MATLAB plottet keine *stetige Funktion* sondern interpoliert ihm punktweise gegebene Funktionen, d.h. man übergibt letztlich Punktmengen $(x(i), y(i))$.

```
1  plot(x,y)
2  % x, y Vektoren der Dimension n
```



Vorgehen beim Plotten

1. Falls die zu plottende Funktion nicht schon als Punktmenge vorliegt, muss sie in eine überführt werden.
   Dazu legt man die gewünschten Intervallgrenzen $[a, b]$ fest und wertet die Funktion an ausreichend vielen Stellen in diesem Intervall aus.
2. Die Punktmenge wird an die Plotfunktion zum Plotten übergeben. MATLAB erzeugt (je nach Einstellung sichtbar oder nicht) eine Grafik.
3. Nach Wunsch können an dem Plot noch verschiedene Einstellungen vorgenommen werden.
4. Falls gewünscht, kann der Plot auch als Grafik in eine Datei abgespeichert werden.

Beispiel: Plotten I

```
 1  % Plotte sin(x) fuer x aus [0 , 2pi].
 2
 3  % Wertetabelle fuer Funktion erstellen
 4
 5  % Vektor mit 101 x-Werten erstellen.
 6  % (d.h. Schrittweite 2 * pi / 100)
 7  x = [0 : 2 * pi / 100 : 2 * pi]
 8
 9  % Fuer jeden x-Wert, also jedes x(i)
10  % den sin ausrechnen und
11  % in Elemente f(i) des Vektors f abspeichern.
12  n = length(x);
13  for i = 1 : n
14    f(i) = sin(x(i))
15  end
16
17  % Tabelle: (nur zur Illustration)
18  % Alle x Werte
19  x
20  % zugehoerige Funktionswerte
21  f
```

Beispiel: Plotten II

```
21  % Plotten mit roter Farbe.
22  % Plot erscheint auf dem Bildschirm
23  plot (x, f, 'r');
24
25  % Ueberschrift und Achsenbeschriftungen !!!
26  % (Matlab kann Latex)
27  title('mein erster Plot');
28  xlabel('0 \leq \vartheta \leq 2\pi')
29  ylabel('sin(\vartheta)')
30
31  % aktuellen Plot als png-Bild speichern in Datei.
32  % erstes Argument: Dateiformat
33  % zweites Argument: Aufloesung
34  % (damit Bild nicht zu gross wird)
35  % letztes Argument: Dateiname
36  print ('-dpng', '-r100', 'sin_img.png');
37
38  % akt. Plot als Postscript-Datei abspeichern.
39  % erstes Argument: Dateiformat (farbiges eps)
40  % letztes Argument: Dateiname
41  print ('-depsc', 'sin_img.eps');
```

Bemerkungen zum Plotten I

1. Beschriften Sie Ihre Plots!!
   ```
   title('So geht eine Überschrift');
   xlabel('0 \leq \vartheta \ leq 2\pi')
   ylabel('sin(\vartheta)')
   ylabel('sin(\vartheta)')
   ```
   MATLAB beherrscht übrigens auch ein paar LaTeX-Befehle (die Wörter mit den \-Zeichen).
2. Die Einstellungen zu einem Plot (wie Überschriften) werden nach dem Plotten angegeben. (Der Plot muss schon da sein, um ihn ändern zu können.)
3. Verwenden Sie die erste Version von `print`:
   ```
   print ('-dpng', '-r100', 'sin_img.png');
   ```
   (außer, Sie wissen genau, was Sie wollen).

Bemerkungen zum Plotten II

- Mehrere Abbildungen in ein Schaubild:
  - 1.Möglichkeit:

```
1  % x1, y1 Vektoren der Dimension n
2  % x2, y2 Vektoren der Dimension m
3  plot(x1,y1,x2,y2)
4  legend('erste Linie','zweite Linie');
```

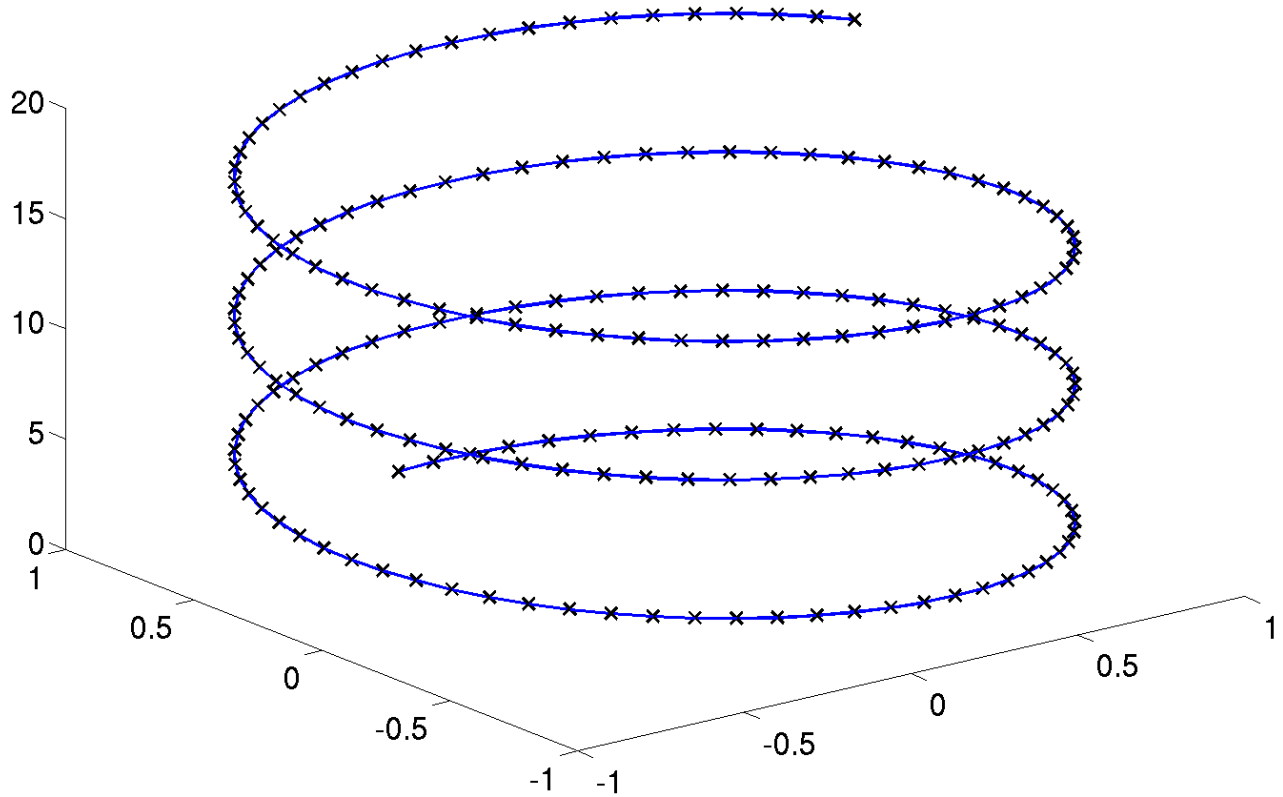
  - 2.Möglichkeit:

```
1  % x1, y1 Vektoren der Dimension n
2  % x2, y2 Vektoren der Dimension m
3  plot(x1,y1)
4  legend('erste Linie');
5  hold on % Garantiert dass nochmal in
6  % dieselbe Figure geplottet wird
7  plot(x2,y2)
8  legend('zweite Linie');
```

Plotten in Matlab: 3D Plot

- 3D-Plot:

```
1  plot3(x,y,z)
2  % x, y, z Vektoren der Dimension n
```



Plotten in Matlab: 3D Plot

- 3D-Plot: $f : [x_1, x_2] \times [y_1, y_2] \to \mathbb{R}$

```
1  x=-1:0.05:1; % Diskretisierung [x_1 ,x_2]
2  y=-1:0.05:1; % Diskretisierung [y_1 ,y_2]
3  [xx,yy] = meshgrid(x,y); % Generiert Gitter
4  zz = xx.^2 - yy.^2; % f(x,y)=x^2-y^2
5  mesh(xx,yy,zz)
```