

MATLAB verfügt über eine nahezu unüberschaubare Anzahl von Möglichkeiten Grafiken zu erzeugen, formatieren oder animieren. In diesem Kapitel werden wir die grundlegenden Werkzeuge für Grafiken in zwei- und drei Raumdimensionen kennenlernen.

Einleitung

*Hinweis:* Beim Lösen der unten folgenden Aufgaben sind teilweise mehrere Zeilen an MATLAB-Funktionen erforderlich. Es wäre also vorteilhaft, die Aufgaben in m-Files (siehe Kap. 1.4) zu lösen.

## 2.1 Zwei-dimensionale Grafiken

2-d

### 2.1.1 Der Befehl plot

Die Grundfunktion zum Erzeugen von 2d-Grafiken heisst

```
plot(...)
```

plot

In den meisten Fällen wird man diese Funktion mit mindestens zwei Angaben verwenden: Einen Vektor mit den x-Koordinaten der Punkte und einen Vektor mit den y-Koordinaten.

*Hinweis:* Die beiden Vektoren müssen die gleiche Länge haben!

**Beispiel:**

```
x = [1 4 5 8 12]
y = [-3 2 5 7 2]
plot(x,y)
```

Man erkennt, dass die y-Werte mit Geraden verbunden werden (→ Polygonzug). Wählt man viel kleinere Schrittweiten, so wird man die Polygonzüge fast nicht mehr erkennen.

**Beispiel:**

```
x = 0:0.01:10;
y = x.^3-8*x.^2;
plot(x,y)
```

**Aufgabe:** Was kommt raus, wenn für den Befehl `plot` nur ein Argument verwendet wird? Machen Sie selbst ein Beispiel dazu.

### 2.1.2 Formatierung der Grafiken

Die Plots können in MATLAB auf vielfältige Weise formatiert werden. Einige Beispiele dazu:

Formatierung

i) Titel, Achsenbeschriftung und Netz:

```
t = 0:0.01:2*pi;
y = sin(t);
plot(t,y);
grid on                                %Netz einschalten
title('Sinus-Funktion')                %Titel
xlabel('Zeit t')                       %x-Achsen Beschriftung
ylabel('sin(t)')                       %y-Achsen Beschriftung
legend('')                             %Legende
```

## ii) Farben und Liniendicke:

```
t = 0:0.001:4;  
y = exp(-t.^2/2);  
  
plot(t,y,'Color','red','LineWidth',3); %rote Linie mit Dicke 3
```

alternativ kann man auch schreiben:

```
plot(t,y,'r','LineWidth',3);
```

<b>r</b>	red
<b>g</b>	green
<b>b</b>	blue
<b>c</b>	cyan
<b>m</b>	magenta
<b>y</b>	yellow
<b>k</b>	black
<b>w</b>	white

**Tab.:** Die wichtigsten Farbabkürzungen

*Hinweis:* Das sind natürlich nicht alle Farben, die mit MATLAB dargestellt werden können. Farben werden mit einem 3-elementigen Spaltenvektor  $[r \ g \ b]$  codiert, mit  $0 \leq r, g, b \leq 1$ , wobei

r: Rotanteil  
g: Grünanteil  
b: Blauanteil

Farbcodierung

(Weiss:  $[0 \ 0 \ 0]$ , Schwarz:  $[1 \ 1 \ 1]$ , Rot:  $[1 \ 0 \ 0]$ , ....)

MATLAB-Beispiel:

```
plot(t,y,'Color',[0.23 0.51 0.92]);
```

**Aufgabe:** Suchen Sie in der Hilfe nach dem Befehl `plot` und versuchen Sie die dargestellten Beispiele zu verstehen.

### 2.1.3 Mehrere Kurven in einem plot darstellen

Es gibt viele Möglichkeiten mehrere Kurven in einer Grafik darzustellen. Dazu betrachten wir ein Beispiel.

**Beispiel:** Darstellen von  $f_1(x)=\sin(x)$  und  $f_2(x)=\cos(x)$  in einem plot für  $x=[0,10]$

Mehrfachplots

Definieren der Vektoren:

```
x = 0:0.05:10;  
f1= sin(x);  
f2= cos(x);
```

#### 1. Möglichkeit:

```
plot(x,f1,x,f2)
```

#### 2. Möglichkeit:

```
plot(x,[f1 ; f2]) Weshalb funktioniert das mit [f1 f2] nicht?
```

## 3. Möglichkeit:

Wenn Sie

```
plot(x, f1);  
plot(x, f2);
```

eingeben, wird nur der zweite Plot dargestellt; der erste wird überschrieben. Mit dem Befehl `hold on` kann man zu einem aktuellen Plot eine zusätzliche Kurve hinzufügen.

```
plot(x, f1);  
hold on  
plot(x, f2);  
hold off (immer mit hold off abschliessen)
```

Mit `legend('Sinus', 'Cosinus')` können Sie dem Plot eine Legende anfügen.

## 2.2 Drei-dimensionale Grafiken

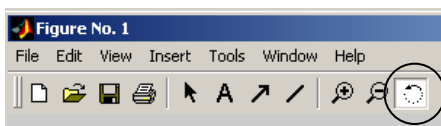
### 2.2.1 Linien in 3d mit `plot3`

Der Befehl `plot3` wird fast gleich wie der Befehl `plot` eingesetzt, mit dem Unterschied, dass man eine zusätzliche Variable (Vektor) braucht.

**Beispiel:**

```
x=0:0.1:10*pi;  
y=exp(-x/20).*cos(x);  
z=exp(-x/20).*sin(x);  
plot3(x,y,z); grid  
xlabel('x');  
ylabel('y');  
zlabel('z');
```

*Hinweis:* Im Grafikfenster kann mit *Rotate 3D* die Grafik rotiert werden.



### 2.2.2 Generieren von Netzen und Oberflächen in 3d

Eine Fläche in 3d ist meistens gegeben durch die Form  $z=f(x,y)$ . D.h jedem Paar  $(x,y)$  wird ein Funktionswert  $z$  zugeordnet. Wir betrachten die allgemeine Vorgehensweise in MATLAB anhand eines Beispiels.

**Beispiel:** Wir wollen die Funktion  $z = \sin(x) \cdot \cos(y)$  für  $0 \leq x \leq 5$  und  $0 \leq y \leq 2$  plotten.

*1. Schritt:* Generieren der x und y-Vektoren

```
x=0:0.05:5;  
y=0:0.02:2;
```

Legende

3d

plot3

## 2. Schritt: Generieren eines Netzes

Die oben definierten Vektoren definieren nur die einzelnen Achsen  $x$  und  $y$ . Nun muss noch die durch die Achse aufgespannte Fläche mit einem rechteckigen Gitter diskretisiert werden. Dazu gibt es in MATLAB den einfachen Befehl `[X,Y]=meshgrid(x,y)`, der aus den definierten Achsen  $x$  und  $y$  ein Gitter erzeugt, das als Matrizen  $X$  und  $Y$  gespeichert wird. (Für das Folgende ist es nicht erforderlich, dass Sie die Struktur der erzeugten Matrizen  $X$  und  $Y$  verstehen!)

```
[X,Y]=meshgrid(x,y);
```

## 3. Schritt: Definieren der Funktion

Mit den erzeugten Matrizen  $X$  und  $Y$  kann nun genau gleich gerechnet werden, wie mit den Vektoren  $x$  und  $y$ . Mit den Unterschied, dass die Funktion nicht nur auf den Achsen ausgewertet wird.

```
Z = sin(X).*cos(Y);
```

## 4. Schritt: Plotten der Funktion

In MATLAB gibt es unzählige Befehle zur Darstellung von Oberflächen. Versuchen Sie die folgenden Befehle mit den oben definierten Matrizen aus:

```
mesh(X,Y,Z)
surf(X,Y,Z)
surfc(X,Y,Z)
contour(X,Y,Z)
```

meshgrid

Fügen Sie bei den Befehlen `surf` oder `surfc` **shading interp** oder **shading flat** hinzu, um die Schattierung der Oberfläche zu verändern:

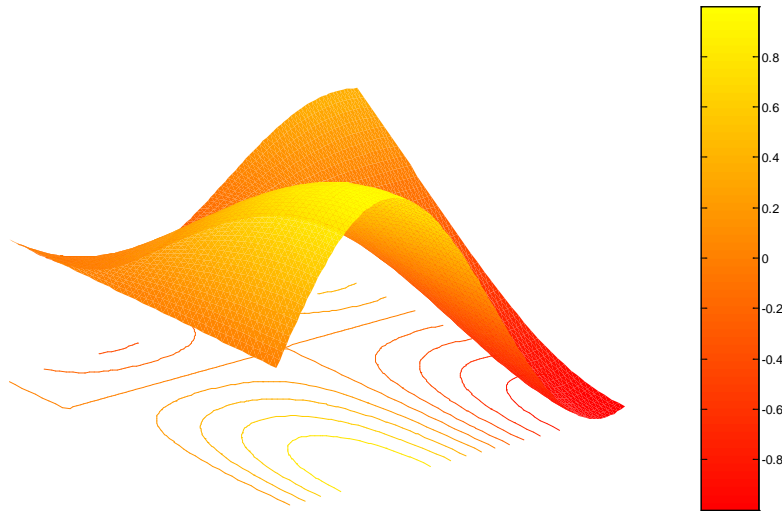
### Beispiel:

```
surfc(X,Y,Z)
shading interp
axis off           %Achsen ausschalten
colormap(pink)     %Farbvorlagen laden (hot, cool, spring,
                  summer, autumn, winter, bone, copper,...)
colorbar           %Farbskala einblenden
```

## 5. Schritt: Exportieren von Grafiken

Eventuell möchten Sie die Grafikdateien in einen Word-Bericht exportieren. Haben Sie mal die Grösse und Orientierung der Grafik festgelegt, können Sie im Figure-Fenster unter *Edit* → *Copy Figure* die Grafik kopieren und danach in einem anderen Programm mit Ctrl+V einfügen.

*Hinweis:* Unter *Edit* → *Copy Options* stehen verschiedene Formate zur Verfügung. Empfehlung für eine gute Qualität: *Preserve information (metafile if possible)* und *Transparent Background*



*Hinweis:* Die unzähligen zusätzlichen Funktionen, Optionen und Formatierungsmöglichkeiten für Grafiken in 2d und 3d können Sie in der MATLAB-Hilfe nachlesen.

### 2.3 Fortgeschrittene Grafikformatierungen mit *Grafik-Handels* (freiwillig)

Die zwei Hauptbefehle dazu sind:

`get`: Dieser Befehl liefert die Eigenschaften eines Objektes

`set`: Damit können die Eigenschaften eines Objektes gesetzt werden

Typische MatLab-Objekte sind Figures, Achsen, Kurven und Texte. Damit diese Objekte formatiert werden können, müssen den Objekten Variablen zugeordnet werden.

**Beispiel:** Plot mit drei Punkten und einem Text an der Stelle (2,6)

```
P1=plot([1 2 3],[1 4 9])
T1=text(2,6,'Text')
```

Gibt man nun im Command-Window `get(P1)` ein, so erscheint eine umfangreiche Liste aller Objekteigenschaften zum Plot, die formatiert werden können. Das Gleiche gilt für den Text `get(T1)`. Mit `set(P1)` und `set(T1)` können dann die gewünschten Eigenschaften formatiert werden. So zum Beispiel

```
set(P1,'LineWidth',2)
set(P1,'xData',[1 2 5])
set(T1,'FontName','Times')
set(T1,'Rotation',50)
set(T1,'FontSize',20)
```

Das Figure-Fenster können auch mit `gcf` angesprochen werden ('get current figure') und die Achsen eines Plots mit `gca` ('get current axis'). Möchte man als Beispiel einen weissen Hintergrund der geöffneten Figure, so lautet der Aufruf:

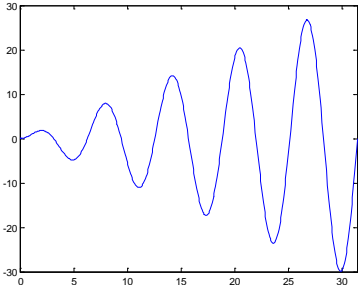
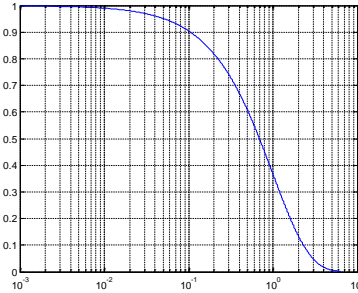
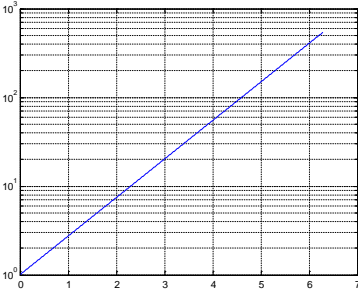
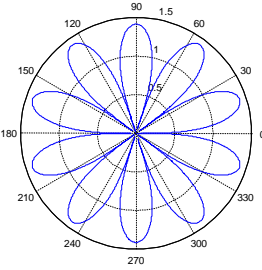
```
set(gcf,'Color','white')
```

oder Farbe als RGB-Codierung [Rot Grün Blau]:

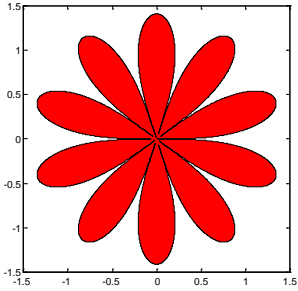
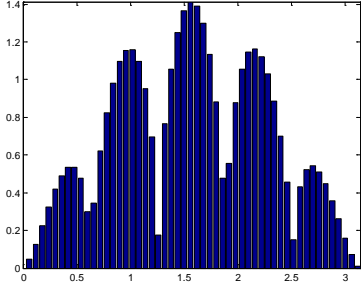
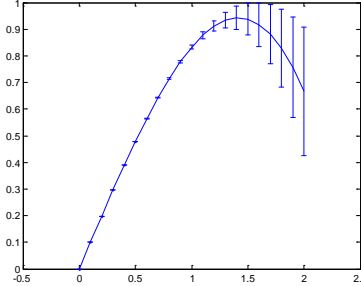
```
set(gcf,'Color',[1 1 1])
```

Figures und Achsen können auch vor dem plot mit `figure` resp. `axis` erzeugt werden. Man kann diese Objekte nun direkt einer Variable zuordnen und einfach ansprechen. `gcf` und `gca` werden damit überflüssig. Diese Methode bietet sich an, falls mit mehreren Figures oder Subplots gleichzeitig gearbeitet werden soll.

## 2.4 Spezielle Grafikfunktionen

Funktion	Beispielskript / Kurzbeschreibung	Ausgabe
<code>fplot</code>	<pre>fplot('x.*sin(x)', [0 10*pi])</pre> <p>(Plot beliebiger Funktion zwischen definierten Grenzen)</p>	
<code>semilogx</code>	<pre>t=0:0.001:2*pi; y=exp(-t); semilogx(t,y); grid</pre> <p>(halblog.Plot: logarithm. x-Achse)</p>	
<code>semilogy</code>	<pre>t=0:0.001:2*pi; y=exp(t); semilogy(t,y); grid</pre> <p>(halblog.Plot: logarithm. y-Achse)</p>	
<code>polar</code>	<pre>t=0:0.001:2*pi; r=sqrt(abs(2*sin(5*t))); polar(t,r);</pre> <p>(plot in Polarkoordinaten)</p>	

## MATLAB: Kapitel 2 – Grafiken

fill	<pre>t=0:0.001:2*pi; r=sqrt(abs(2*sin(5*t))); x=r.*cos(t); y=r.*sin(t); fill(x,y,'red') axis('square')</pre> <p>(gefüllte 2-D Plots)</p>	
bar	<pre>t=0:0.06:2*pi; r=sqrt(abs(2*sin(5*t))); y=r.*sin(t); bar(t,y); axis([0 pi 0 inf]);</pre> <p>(Balkendiagramm Plot)</p>	
errorbar	<pre>x=0:0.1:2; approx=x-x.^3/6; er=approx-sin(x); errorbar(x,approx,er)</pre> <p>(“Vertrauensintervall” entlang Kurve darstellen)</p>	

### Testataufgaben Kapitel 2:

Diese Aufgaben gehören zu den Testatbedingungen und sind bis Mittwoch, 27. April 2016, auf Ilias in den Ordner

> Briefkasten > Abgabe Matlab > Serie2

als hochzuladen. Packen Sie **alle Übungen in ein File (m-File oder pdf)**. Testate die zu spät hochgeladen werden, werden nicht berücksichtigt.

### Übung 2.1: 2d-Plots

- a) Plotten Sie die Funktion  $y=y(x)$  gegeben durch die Vektoren

$x=[0 \ 1 \ 3 \ 6 \ 10]$  und  $y=[2 \ 3 \ 0 \ -1 \ 0]$

Verbinden Sie die Punkte mit gestrichelten Linien und markieren Sie die Punkte mit Kreisen oder Quadraten. Schauen Sie dazu in der Hilfe nach!

- b) Stellen Sie die drei Funktionen

$$f_1(x) = x \quad f_2(x) = x^2 \quad f_3(x) = x^3$$

im Intervall  $0 \leq x \leq 2$  in einem plot dar. Verwenden Sie drei verschiedene Farben und fügen Sie eine Legende hinzu. Erzeugen Sie mit `set(...)` einen weissen Hintergrund.

- c) Zeichnen Sie mit Hilfe von `plot` die Kontur eines beliebigen Dreiecks. Machen Sie nun das Gleiche mit der Funktion `fill`.

### Übung 2.2: Import von Daten aus Dateien

Im Ordner MATLAB in Ilias ist die Datei `data.txt` abgelegt. In der ersten Spalte befinden sich die x-Werte und in der zweiten Spalte die y-Werte aus einer beliebigen Messung.

Speichern Sie dieses File lokal ab und schreiben Sie ein m-File, das zuerst die Daten einliest und anschliessend grafisch (in einem 2d Plot) ausgibt.

Verwenden Sie dazu den Befehl `importdata(...)`

### Übung 2.3: 3d-Plots

- a) Stellen Sie die Ebene gegeben durch  $f(x, y) = 3 - 2x + y$  mit `surf(...)` dar mit  $-3 \leq x \leq 3$  und  $0 \leq y \leq 5$ . Beschriften Sie die Achsen und geben Sie einen Titel an.
- b) Zeichnen Sie die beiden Funktionen  $f_1(x, y) = x^2 + y^2$  und  $f_2(x, y) = 6\sin(x)^2 \cdot \cos(y)$  in einer Grafik für  $-2 \leq x \leq 2$  und  $-2 \leq y \leq 2$ . Verwenden Sie dazu die Befehle:

```
surf
hold on
axis off
shading interp
colormap(..)
colorbar
alpha(..)
```

Schauen Sie in der Hilfe nach, falls Sie einen Befehl nicht verstehen, so zum Beispiel durch die Eingabe `>>doc alpha` (im Command Window)

### Übung 2.4: (Matrizen als lineare Abbildungen)

*MatLab-Hinweis:* Sie können mit dem Befehl `subplot(...)` mehrere Achsen in einem Figure-Fenster darstellen. Geben Sie mal den folgenden Code ein und beobachten Sie, was dabei passiert:

```
subplot(211)
```



## MATLAB: Kapitel 2 – Grafiken

```
x=0:0.1:5;    y=sin(x);    plot(x,y);
```

```
subplot(212)  
x=2:0.1:5;    y=x.^3;    plot(x,y);
```

Schauen Sie sich in der Hilfe andere Beispiele zum Befehl **subplot** an.

- a) Zeichnen Sie mit der Funktion `fill` ein Quadrat mit den Eckpunkten (`subplot(121)`)

$$p_1 = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \quad p_2 = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad p_3 = \begin{pmatrix} 1 \\ 1 \end{pmatrix} \quad p_4 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

Definieren Sie zur besseren Darstellung die Achsen als `axis([-3 3, -3 3])`

- b) Transformieren Sie nun diese Punkte mit verschiedenen Matrizen  $A$  (Multiplikation) und zeichnen Sie das „deformierte“ Quadrat in einen neuen Plot (`subplot(122)`)

$$A = \begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} 3 & 0 \\ 0 & 3 \end{pmatrix}, \begin{pmatrix} 2 & 2 \\ 2 & 2 \end{pmatrix}, \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} 1 & -1 \\ 0 & 1 \end{pmatrix}, \text{ oder } \begin{pmatrix} \cos(t) & -\sin(t) \\ \sin(t) & \cos(t) \end{pmatrix} \text{ mit } 0 \leq t \leq 2\pi$$

oder erfinden Sie selbst was!