WiMa-Praktikum 1

Universität Ulm, Sommersemester 2019

Woche 5

Lernziele

In diesem Praktikum sollen Sie üben und lernen:

- Einfache 2D Plots mit `plot`
- Figurensteuerung und mehrere Plots in einem Bild
- Wichtige Plot-Eigenschaften, Plotbearbeitung und -beschriftung
- Logarithmische Plots
- 3D Plots mit `plot3`, `mesh` und `surf`

Am Anfang geben wir Ihnen einen kurzen Überblick über die benötigten MATLAB-Anweisungen. Beantworten Sie danach bitte erst einige Fragen, bevor Sie sich an den Rechner setzen!

Bilder in Matlab

In MATLAB können Ergebnisse im zugehörigen Grafiksystem visualisiert werden. Eine gute Übersicht über die Visualisierungsmöglichkeiten in MATLAB einschließlich der möglichen 2D- oder 3D-Plot-Typen bietet die Seite *Graphics* der MATLAB-Dokumentation: `https://de.mathworks.com/help/matlab/graphics.html`. Einige der Plot-Befehle werden wir im Folgenden kennen lernen.

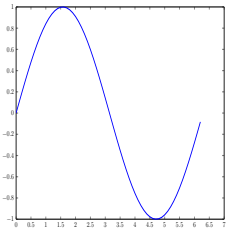
2D Linienplot `plot`

Mit dem Befehl `plot` lässt sich in Matlab ein zweidimensionaler Graph erzeugen. Für zwei Vektoren `x` und `y` kann man

`plot(y)` oder `plot(x,y)`

verwenden. Der erste Befehl plottet den Vektor `y`, wobei der i-te Eintrag y_i des Vektors den y-Wert zum Punkt i auf der x-Achse darstellt. Hingegen plottet der zweite Befehl den Vektor `y` gegen `x`. Es werden also alle Punkte (x_i, y_i) in ein 2D-Koordinatensystem eingezeichnet und mit einer Linie verbunden. Hierbei müssen die Vektoren `x` und `y` gleich lang sein, ansonsten kann `plot` keine eindeutige Zuordnung vornehmen.

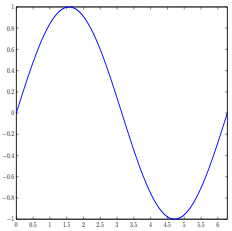
```
>> x = 0:0.1:2*pi;
>> y = sin(x);
>> plot(x,y)
```



Das selbe Bild erzeugt der Befehl `fplot`, welcher Funktionen direkt plottet. Die Syntax lautet:

`fplot('Funktion', Definitionsbereich).`

```
>> fplot('sin', [0, 2*pi])
```



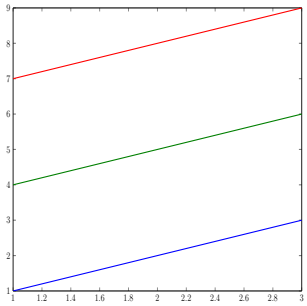
Der Befehl `plot(x,y)` kann auch für zwei Matrizen `x` und `y` (der selben Dimension) verwendet werden. Es werden dann die jeweiligen Spalten gegeneinander geplottet, so dass mehrere Kurven in einem Plot dargestellt werden. Mehrere Kurven in einem Plot können auch durch Benutzen der folgenden Syntax dargestellt werden:

`plot(x1,y1,...,xn,yn)`

Es werden die unterschiedlichen x-y-Paare in einem Graphen mit denselben Achsen dargestellt. Hierbei können die Vektorpaare jeweils von unterschiedlicher Länge sein.

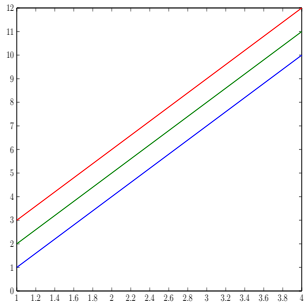
Sofern alle Vektoren `x1`, ..., `xn` und `y1`, ..., `yn` gleich lang sind, also $\dim(\texttt{x1}) = \ldots = \dim(\texttt{xn}) = \dim(\texttt{y1}) = \ldots = \dim(\texttt{yn})$, ist der Befehl identisch zu `plot(X,Y)` mit `X = [x1,...,xn]` und `Y = [y1,...,yn]`, wobei `x1,...,xn,y1,...,yn` jeweils Spaltenvektoren sein müssen.
Wie man im folgenden Beispiel sehen kann, muss bei der Benutzung einer Datenmatrix unbedingt auf die Orientierung der Vektoren geachtet werden - Matlab plottet grundsätzlich spaltenweise, solange die Spalten-Dimension der y-Matrix zur Dimension des x-Vektors passt.

```
>> x1 = 1:3;
>> y1 = [1; 2; 3];
>> y2 = [4; 5; 6];
>> y3 = [7; 8; 9];
>> plot(x1,[y1, y2, y3])
```



Fügt man beim x-Vektor ein viertes Element und bei der y-Matrix eine vierte Spalte hinzu, passt die Zeilenanzahl der y-Matrix (3 Zeilen) nicht mehr zur Länge des Vektors x und MATLAB plottet zeilenweise.

```
>> x2 = 1:4;
>> y4 = [10;11;12];
>> plot(x2,[y1, y2, y3, y4])
```



Linienart, Farbe und Marker

Manchmal bietet es sich an, keine durchgezogenen Linien bei der Darstellung zu verwenden. Zum Beispiel um deutlich zu machen, dass es sich um eine Approximation an endlich vielen Punkten handelt. Stattdessen kann entweder eine unterbrochene Linienart verwendet und/oder die dargestellten Punkte durch Marker gekennzeichnet werden. Die Linienart, der Markertyp aber auch der Farbtyp können bei `plot` leicht verändert werden:

```
plot(x, y, LineSpec)
```

Zum Beispiel erzeugt `'*'` für `LineSpec` Sternchen an den jeweils dargestellten Punkten, die Punkte werden dann nicht mehr mit einer Linie verbunden und die Farbe wird von MATLAB festgelegt. Verwendet werden können unter anderem folgende Befehle

line type		color type		marker type	
-	durchgezogen	r	rot	*	Sternchen
	(default)	g	grün	.	Punkte
--	gestrichelt	b	blau	+	Kreuze
:	feingestrichelt	c	türkis (cyan)	s	Quadrat (square)
-.	Punkt-Strich	m	magenta	<,>	Dreieck
		k	schwarz (karbon)		nach links, rechts

Der string `LineSpec` kann (mit bis zu vier Zeichen) als Kombination der drei Typen definiert werden, es kann aber genau so gut nur einer oder zwei der drei Typen festgelegt werden.

```
>> x = 1:0.1:2*pi
>> plot(x, sin(x), 'r-.*')
```

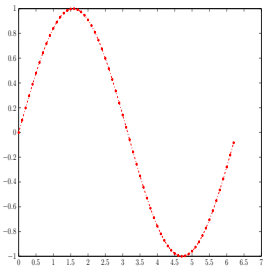
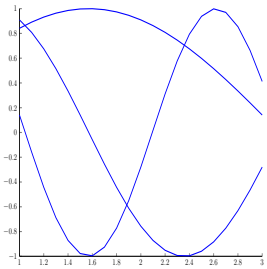


Figure-Steuerung, mehrere Plots in einem Schaubild, Subplots

Bisher wurde das MATLAB-Schaubild automatisch geöffnet. Bei Eingabe eines weiteren Plot-Befehls wird der alte Plot im selben Schaubild automatisch überschrieben. Um dies zu verhindern, gibt es den Befehl `figure`. Mit Eingabe von `figure` lässt sich manuell ein neues Fenster öffnen. Dieses ist dann aktiv und der Plot-Befehl wird immer im aktiven Fenster ausgeführt. Die Fenster werden automatisch durchnummeriert. Man kann diese Nummerierung mit `figure(Zahl)` selber vornehmen oder, sofern das z. B. 3. Fenster schon existiert, es mit `figure(3)` direkt ansteuern. Der Befehl `clf` löscht im Übrigen alle Grafikobjekte des aktiven Schaubilds, `clf('reset')` löscht zusätzlich auch noch alle Schaubild-Eigenschaften.

Mehrere Plots übereinander gelegt in einem Schaubild kann mit `hold` realisiert werden. Ab Eingabe von `hold on` werden alle folgenden Plots in das selbe Schaubild gezeichnet, bis mit `hold off` der Vorgang beendet wird.

```
>> figure
>> hold on
>> for k = 1:3
>>     plot(1:0.1:3, sin(k*(1:0.1:3)))
>> end
>> hold off
```

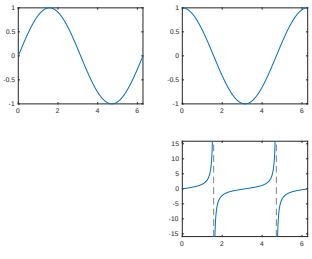


Zusätzlich können mehrere Schaubilder in einem Bild gezeichnet werden. Mit

```
subplot(Z, S, N)
```

werden in `Z` Zeilen und `S` Spalten `Z*S` Plots dargestellt, die mit `N` durchnummeriert werden.

```
>> subplot(2,2,1);
>> fplot('sin', [0,2*pi])
>> subplot(2,2,2);
>> fplot('cos', [0,2*pi])
>> subplot(2,2,4);
>> fplot('tan', [0,2*pi])
```



Beschriftungen und Ploteigenschaften

Durch Benutzen des Ploteditors können Plots interaktiv angepasst werden. Das Öffnen des Ploteditors erfolgt entweder mit Hilfe des Befehls `plottools`, durch Doppelklick im Plot auf z. B. die zu verändernde Linie oder durch Verwendung des Menüs des MATLAB-Figure-Fensters. Es ist jedoch meist komfortabler und sinnvoll, die Eigenschaften des Schaubilds direkt beim `plot`-Befehl im MATLAB-Programm mit den passenden MATLAB-Befehlen festzulegen. Die wichtigsten werden im Folgenden vorgestellt.

Zunächst können die Achsen mit

```
axis([xmin, xmax, ymin, ymax])
```

angepasst werden. Die Eingabe von `xlim` oder `ylim` gibt die aktuell verwendeten Werte zurück. Weiter fixiert `xlim manual` die verwendeten Grenzen der x-Achse, so dass diese im Folgenden z. B. bei der Nutzung von `hold` nicht mehr angepasst werden. Mit `xlim auto` passt MATLAB die Achsen wieder an die Daten an.

Zusätzlich können die Legende, die Achsenbeschriftung und ein Titel des Schaubilds festgelegt werden.

Befehl	**Bedeutung**
`title`	Titel, mittig über Plot
`xlabel, ylabel`	Beschriftung von x-, y-Achse
`legend`	Beschriftung der Kurven

Eigenschaften wie Schriftgröße, Schriftart oder welche Punkte auf der x- oder y-Achse dargestellt werden sollen, werden am einfachsten über den *current axis handle* `gca` gesteuert. Mit `gca` werden die Achsen der aktiven Figur angesteuert und über den Befehl

```
set(gca, 'Name', Wert)
```

können diese bearbeitet werden. Folgende Eigenschaften können unter anderem verändert werden:

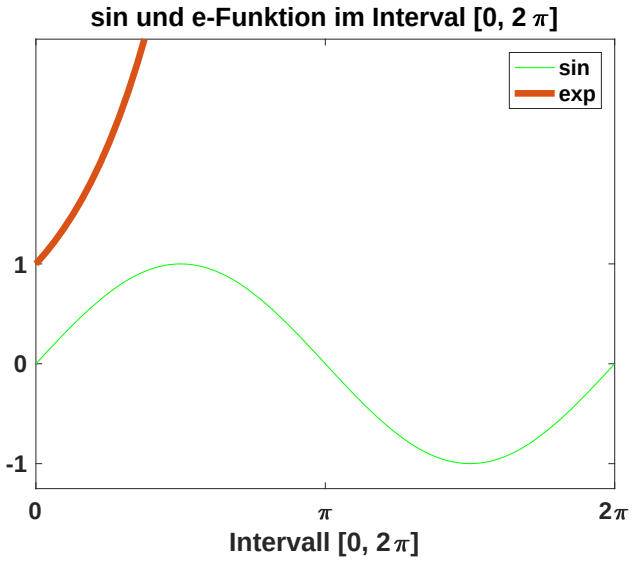
Name	**Bedeutung**
FontSize	Schriftgröße
FontName	Schriftart
FontWeight	Schriftstil
XTick, YTick	Punkte auf der x,y-Achse
XTickLabel	Beschriftung der Punkte an der X-Achse

Über `get(gca)` zeigt MATLAB eine komplette Liste aller veränderbarer Eigenschaften mit ihren aktuellen Werten an.

Das folgende Beispiel-Skript `ex5x1` plottet zunächst die Sinusfunktion im Intervall $[0, 2\pi]$ in grün. Es wird der Bildausschnitt $[0, 2\pi] \times [-1.5, 3.25]$ fixiert und mit Hilfe von `hold on` die e-Funktion in die selbe Figur gezeichnet. Der `plot`-Befehl bestimmt eine stärkere Liniendicke für die Kurve der e-Funktion. Durch das Fixieren des Ausschnitts ist nur ein sehr kleiner Teil dieser Kurve sichtbar. Zuletzt bekommt der Plot eine Überschrift und eine Legende.

```
% ex5x1
%
% Plot von Sinus und e-Funktion

fplot('sin', [0,2*pi], 'g')
% Layout
set(gca,'FontSize',18,'FontName','Arial','FontWeight','bold')
% Bildausschnitt
axis([0, 2*pi, -1.25, 3.25])
% Punkte
set(gca, 'XTick', 0:pi:2*pi, 'YTick', -1:1)
set(gca, 'XTickLabel', {'0', '\pi', '2\pi'})
% Achsenausschnitt fixieren
axis manual
% der naechste Plot im selben Bild:
hold on
% e-Funktion in [0,5]
plot(0:0.1:5, exp(0:0.1:5), 'Linewidth', 5)
hold off
% Beschriftung
title('sin und e-Funktion im Interval [0, 2\pi]')
xlabel('Intervall [0, 2\pi]')
% Legende
legend('sin', 'exp')
```



Logarithmische Plots, zwei y-Achsen

Für Fehlerplots bietet sich eine logarithmische Achsenskalierung an. Die folgenden Plot-Befehle haben dieselbe Syntax wie `plot`.

`loglog`	beide Achsen werden logarithmisch zur Basis 10 skaliert
`semilogx, semilogy`	die x- bzw. y-Achse wird logarithmisch zur Basis 10 skaliert

Außerdem können mit

```
plotyy(x1, y1, x2, y2)
```

zwei Kurven mit verschiedenen y-Achsen geplottet werden.

3D-Plots

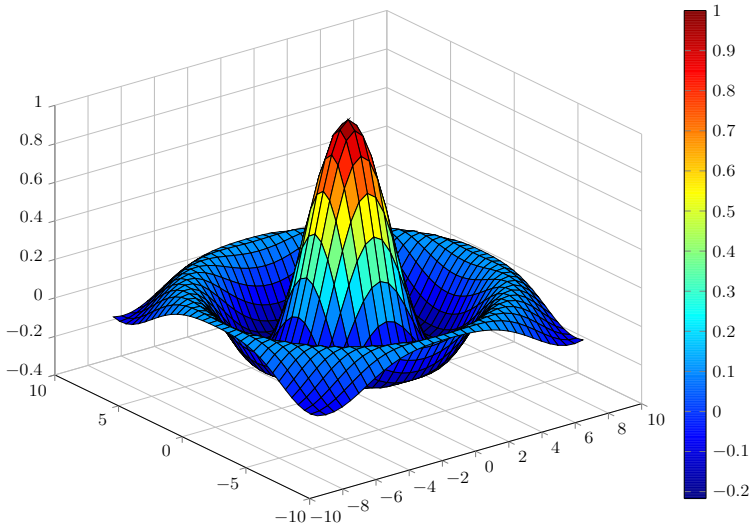
Eine 3D-Erweiterung von `plot` stellt `plot3` dar. Wie bei `plot` lautet die Syntax

```
plot3(x, y, z, LineSpec).
```

Oberflächen können zum Beispiel mit `mesh` oder `surf` dargestellt werden. Man kann jede Matrix A mit `mesh(A)` plotten. Die Matrixeinträge a_{ij} sind dann die z-Koordinaten (und bestimmen gleichzeitig die Farbe) zum x-Wert i und y-Wert j. `mesh` erzeugt ein dreidimensionales coloriertes Gitternetz, `surf` erzeugt eine colorierte Oberfläche.
Konkrete x- und y-Koordinaten müssen als Matrizen angegeben werden. Mit Hilfe von `meshgrid` können die passenden Gittermatrizen erzeugt werden.

```
>> [X,Y] = meshgrid(-2:1:2);
>> X
X =
    -2    -1     0     1     2
    -2    -1     0     1     2
    -2    -1     0     1     2
    -2    -1     0     1     2
    -2    -1     0     1     2
>> Y
Y =
    -2    -2    -2    -2    -2
    -1    -1    -1    -1    -1
     0     0     0     0     0
     1     1     1     1     1
     2     2     2     2     2
```

```
>> [X, Y] = meshgrid(-8:0.5:8);
>> r = sqrt(X.^2 + Y.^2) + eps;
>> Z = sin(r) ./r ;
>> surf(X, Y, Z)
>> colorbar
```



Mit `colorbar` wird die Legende eingeblendet.

Offline Aktivitäten

Übereinstimmen

Schreiben Sie vor jeden Begriff auf der linken Seite den passenden Buchstaben der Beschreibung, die am besten mit der aus der rechten Spalte übereinstimmt.

_____	1.	`set`	a.	Definiert die Linienstärke.
_____	2.	`':'`	b.	Löscht den Plot des aktiven Schaubilds.
_____	3.	`plot`	c.	Plottet Funktionen direkt.
_____	4.	`gca`	d.	Erzeugt ein Koordinaten-Gitter.
_____	5.	`clf`	e.	current axis handle
_____	6.	`plot3`	f.	Hat die selbe Syntax wie `plot`.
_____	7.	`LineWidth`	g.	Erzeugt 2D-Linienplots.
_____	8.	`fplot`	h.	Legt Plot-Eigenschaften fest.
_____	9.	`'b'`	i.	Definiert die Farbe blau.
_____	10.	`meshgrid`	j.	Definiert eine feingestrichelte Linie.

Ihre Antwort:

Füllen Sie die Lücken aus

Ergänzen Sie die folgenden Sätze.

11. Zugriff auf den kleinsten und größten Wert der dargstellten x-Achse erhält man mit _____.
12. Die _____ beschriftet die verschiedenen Kurven im Plot.
13. Matrizen werden mit `plot` wenn möglich _____ geplottet.
14. Der _____ ermöglicht die interaktive Bearbeitung eines Plots.
15. Alle Datenobjekte in einer aufgerufenen Funktion sind _____.
16. Zugriff auf die MATLAB-Dokumentation liefern die Befehle _____ .
17. Der Befehl _____ beendet das Zeichnen mehrerer Plots in ein Bild.
18. Mit `loglog` werden die Achsen _____ zur Basis 10 skaliert.
19. Für `mesh` müssen die Gitterpunkte als _____ angegeben werden.
20. Der _____ definiert, wie die Punkte im Plot gekennzeichnet werden.

Ihre Antwort:

Programmausgaben

Für jedes der folgenden Programmsegmente, lesen Sie zuerst die Zeilen und schreiben Sie die Ausgabe an die dafür vorgesehene Stelle.

21. Welche Farbe hat die geplottete Funktion?

```
1  x = -1:0.1:1;
2  y = x.^2;
3
4  plot(x, y, ':g')
```

Ihre Antwort:

22. In welchem Muster erscheinen die Subplots?

```
1  x = 0:0.1:2*pi;
2
3  subplot(3,3,2);
4  plot(x, sin(x))
5  subplot(3,3,4);
6  plot(x, x.^2);
7  subplot(3,3,5);
8  plot(x, x);
9  subplot(3,3,6);
10 plot(x, x.^2);
11 subplot(3,3,8);
12 plot(x, sin(x));
```

Ihre Antwort:

23. Was wird geplottet?

```
1  x = 0 : 0.001*pi : pi+0.001;
2  hold on
3    plot(cos(x), sin(x), 'k');
4    plot(cos(x), -sin(x), 'k');
5  hold off
```

Ihre Antwort:

Korrigieren Sie den Code

Für jedes der folgenden Codesegmente sollen Sie feststellen, ob ein Fehler enthalten ist. Falls ein Fehler vorliegt, markieren Sie diesen und spezifizieren Sie, ob es sich dabei um einen Semantik- oder Syntaxfehler handelt. Schreiben Sie die korrigierten Anweisungen jeweils in den dafür vorgesehenen Bereich unter der Problemstellung. Bemerkung: Es kann sein, dass ein Programm mehrere oder keine Fehler enthält.

24. Das folgende Matlab-Skript soll die Funktion $f(x) = x^3$ auf $[-3, 3]$ plotten.

```
1  figure
2  plot(x, x^3)
```

Ihre Antwort:

25. Das folgende Skript soll zwei Bilder anzeigen, und zwar jeweils eine Spirale, einmal in 3D und einmal die Projektion der Spirale auf die x-y-Ebene:

```
1  t = 0:pi/50:10*pi;
2  figure(1)
3  plot3(cos(t),sin(t),t)
4
5  figure(1)
6  plot(cos(t),sin(t))
```

Ihre Antwort:

Praktikumsaufgabe - Plotten von Funktionen

Lesen Sie die Aufgabenstellung, studieren Sie dann die vorgegebenen Programmzeilen. Ersetzen Sie dann die %% Kommentare im vorgegebenen Code durch Matlab-Anweisungen und führen Sie das Programm aus.

26. Informieren Sie sich über die Funktion `contour`. Plotten Sie in einem Bild die Gaußfunktion $z = f(x, y) = \exp(-x^2 - y^2)$ zweidimensional sowohl als Konturplot mit Hilfe von `contour` als auch als Pseudocolor-Plot mit `pcolor(x,y,z)`. Die Höhenlinien sollen schwarz sein.

    Stellen sie danach die Funktion dreidimensional dar, als Gitternetz und als Oberfläche.

    Probieren Sie den Ploteditor aus.

27. Zeichnen Sie dreidimensional eine Sprungfeder. Die Spiralfeder soll in der Mitte einen kleinen Radius, oben und unten einen deutlich größeren haben. Benennen Sie den Plot.

28. Wir wollen die Ableitung der e-Funktion an der Stelle 0 approximieren. Dafür betrachten wir für n gegen ∞ den Grenzwert des Differenzenquotienten $\lim_{n\to\infty} \frac{e^{1/n}-e^0}{1/n}$.

    Plotten Sie den Funktionsverlauf des Differenzenquotienten gegen die Werte von n im Intervall $[1, 100]$. Plotten Sie den Fehler zwischen dem Differenzenquotienten und der Ableitung von e^x an der Stelle $x = 0$ mit `plot`, mit `semilogy` und mit `loglog`. Welches Bild zeigt den Fehlerverlauf am deutlichsten?

    Sorgen Sie dafür, dass man die Beschriftung gut lesen kann und beschriften Sie die y-Achse in diesem Bild mit *Fehler*. Benennen Sie das Bild mit *Fehlerverlauf*. Verstärken Sie die Linie auf 2 und färben Sie sie violett (magenta).