Universität zu Köln

PROF. DR. ANGELA KUNOTH
MAX GLEMBOCK, LASLO HUNHOLD

MATLAB PRÄSENZAUFGABEN — WS 2017/2018

Übungsblatt 4

Ausgabe: 8.11.2017

Grafiken mit Matlab erstellen – 2-dimensionale Plots

In Matlab ist es relativ leicht, verschiedenste Arten von Grafiken zu erzeugen. Solche Grafiken können sehr individuell gestaltet werden. Bspw. kann man Farben, Achsenskalierungen und Beschriftungen den eigenen Vorstellungen anpassen. Grundsätzlich gibt es zur Modifikation von Grafiken zwei Möglichkeiten; einerseits direkt über Matlab-Code oder andererseits interaktiv in einer bereits erstellen Grafik. Wir werden uns auf die erste Variante beschränken.
Die einfachste Variante, zweidimensionale Plots zu erzeugen, besteht darin, zwei Vektoren mit gleichen Dimensionen zu koppeln. Hat man zwei Vektoren `x` und `y` gegeben, werden mit dem Befehl `plot(x,y)` die jeweiligen $x(i)$ und $y(i)$ als zusammengehörig erkannt und ein entsprechender Plot erzeugt, in dem die einzelnen $y(i)$ standardmäßig durch einen Polygonzug verbunden sind.
Mit zusätzlichen Angaben kann man das Aussehen des Plots modifizieren. Ein allgemeinerer Aufruf der Funktion `plot` hat die Form `plot(x,y,String)`, wobei sich der String aus bis zu drei Angaben (Farbe, Knoten, Linienart) zusammensetzt. Die möglichen Eingaben können den folgenden Tabellen entnommen werden:

Farbe	
r	Red
g	Green
b	Blue
c	Cyan
m	Magenta
y	Yellow
k	Black
w	white

Marker	
o	Kreis
*	Stern
.	Punkt
+	Plus
x	Kreuz
s	Quadrat
d	Diamant
ˆ	Dreieck nach oben
v	Dreieck nach unten
<	Dreieck nach rechts
>	Dreieck nach links
p	Fünfpunktestern
h	Sechspunktestern

Linienart	
-	durchgezogene Linie
--	gestrichelte Linie
:	gepunktete Linie
-.	gepunktete und gestrichelte Linie

Ein typischer Aufruf hätte die Form `plot(x,y,'mx:')`. Das gleiche Resultat erhält man mit `plot(x,y,'m:x')`, woran man sieht, dass die Reihenfolge der Angaben irrelevant ist.

Aufgabe 14 (Erstellen eines Plots)
Gegeben seien die Vektoren x und y, die wie folgt definiert sind:

$$x = \begin{pmatrix} 1.5, & 2.2, & 3.1, & 4.6, & 5.7, & 6.3, & 9.4 \end{pmatrix}^T \quad \text{und} \quad y = \begin{pmatrix} 2.3, & 3.9, & 4.3, & 7.2, & 4.5, & 3.8, & 1.1 \end{pmatrix}^T.$$

Erstellen Sie einen Plot, in dem jeder x-Stelle der entsprechende y-Wert zugeordnet wird. Die Farbe des Polygonzugs sollte dabei blau, die Linienart gepunktet und der Marker ein Kreis sein. Ist es egal, ob Sie die Vektoren als Zeilen- bzw. Spaltenvektoren initialisieren?

Es ist möglich, mehr als einen Polygonzug in ein Koordinatensystem zu legen. Dazu fügt man dem Argument von `plot` lediglich weitere Vektoren sowie die gewünschten Optionen für diesen Polygonzug an. Ferner können `plot` auch noch weitere Attribute übergeben werden wie bspw. `LineWidth`:

```
plot(x,y,'LineWidth',2)
```

Durch `LineWidth` und den darauffolgenden natürlichen Zahlenwert wird die Dicke **aller** Polygonzüge und Marker in einem Plot festgelegt.

Aufgabe 15 (Erstellen eines Plots – 2)
Erweitern Sie den Plot aus Aufgabe 14 um eine Darstellung der harmonischen Folge ($a_n = \frac{1}{n}, n \geq 1$) an den Auswertungsstellen $n = 1, \ldots, 10$. Die Auswertungen sollen als grüne Diamanten verdeutlicht werden und es sollen keine Verbindungslinien zwischen den Auswertungen sichtbar sein. Legen Sie zusätzlich die Liniendicke auf 4 fest.

Bemerkung: In Matlab ist es nicht möglich, einen Graphen stetig darzustellen. Auch bei der Visualisierung von anonymen Funktionen, die wir auf dem nächsten Blatt besprechen, wird immer mithilfe einer endlichen Anzahl von Stützstellen gearbeitet und es werden Polygonzüge dargestellt. Dennoch werden wir im Folgenden die Begriffe „Graph“ und „Polygonzug“ synonym verwenden.

In Matlab werden nicht nur Vektoren als „Datenquellen“ akzeptiert, sondern auch Matrizen. Wenn `x` ein m-dimensionaler Vektor und `Y` eine $m \times n$-Matrix ist, bewirkt `plot(x,Y)` das gleiche wie `plot(x,Y(:,1),x,Y(:,2),...,x,Y(:,n))`. Wenn zusätzlich `X` auch eine $m \times n$-Matrix ist, kann man mit `plot(X,Y)` die jeweils korrespondierenden Spalten der beiden Matrizen zu einem Graph verbinden; es handelt sich also um eine verkürzte Schreibweise von `plot(X(:,1),Y(:,1),...,X(:,n),Y(:,n))`.

Für viele Anwendungen ist es sinnvoll, die Achsen logarithmisch zu skalieren. Dazu verwendet man statt `plot` eine der folgenden Anweisungen:

- `semilogx(x,y)`: Die x-Achse wird logarithmisch skaliert.
- `semilogy(x,y)`: Die y-Achse wird logarithmisch skaliert.
- `loglog(x,y)`: Beide Achsen werden logarithmisch skaliert.

Matlab bietet für die Formatierung eines Plots bzw. des `figure`-Fensters zahlreiche weitere Optionen an. Wichtige Befehle sind z.B. `xlabel(`*String*`)` bzw. `ylabel(`*String*`)`, die der x-Achse bzw. der y-Achse eine Beschriftung verleihen. Der Befehl `title(`*String*`)` verschafft dem Plot eine Überschrift.
Ein weiterer nützlicher Befehl ist `hold on`. Dieser sorgt dafür, dass beim Plotten von weiteren Funktionen die bisher Erstellten im `figure`-Fenster erhalten bleiben und nicht – wie standardmäßig eingestellt bzw. bei `hold off` – zunächst das `figure`-Fenster „geleert“ und erst dann der neue Plot eingefügt wird.
Wenn man mehrere `figure`-Fenster ausgegeben haben möchte, so muss mit dem Befehl `figure` erst ein neues Fenster generiert werden. Mit `figure(n)` kann man ein beliebiges `figure`-Fenster aktivieren (jeweils das `n`-te, was erzeugt wurde), so dass sich alle folgenden (Grafik-)Befehle auf dieses beziehen.

Aufgabe 16 (Logarithmische Skalierung eines Plots)
Visualisieren Sie den Fehler einer quadratischen Taylor-Approximation der Exponentialfunktion im Nullpunkt bei kleinen Zahlen und zeigen Sie, dass dieser sich proportional zu h^3 verhält. Betrachten Sie dazu $|1 + h + \frac{h^2}{2} - \exp(h)|$ und zum Vergleich h^3 jeweils ausgewertet für $h = 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1$ und mit jeweils logarithmischer Skalierung auf beiden Achsen.
Fügen Sie zusätzlich geeignete Achsenbeschriftungen sowie einen Titel hinzu. Geben Sie Ihr Ergebnis mithilfe des `hold on`-Befehls in **einer** Grafik aus.

Viele Eigenschaften der Achsen können über den `axis`-Befehl eingestellt werden. Mögliche Befehle sind:

- `axis ([xmin xmax ymin ymax])`: Erlaubt die manuelle Festlegung der Achsenabschnitte.
- `axis auto`: Automatisiert die Festlegung der Achsenabschnitte (Standardeinstellung).
- `axis equal`: Legt die gleichen Achsenabschnitte für x-Achse und y-Achse fest.
- `axis square`: Gibt einen Plot in quadratischer Darstellung zurück.
- `axis tight`: Passt die Achsengrenzen an die darzustellenden Werte an.

Für den Fall, dass man nur die Grenzen einer Achse festlegen möchte, stehen die Befehle `xlim([xmin, xmax])` und `ylim([ymin, ymax])` zur Verfügung. Sollte man sich an einem Ende des Intervalls nicht für eine Grenze festlegen wollen, sorgt die Angabe von `-inf` bzw. `inf` dafür, dass Matlab sich darum kümmert.

Wir haben bereits gesehen, wie man seinem Plot einen Titel verleiht, doch kann man in einem Plot Text auch an einer beliebigen Stelle einfügen und sich sogar einer automatisch erstellten Legende bedienen. Um Text einzufügen, reicht der Befehl `text(x,y,`*String*`)`, wobei (x, y) die Koordinaten im Plot festlegt, an denen der Text beginnen soll. Das Kommando zum Einfügen der Legende lautet `legend(`*String1*`, ..., `*Stringn*`, position)`. Die Reihenfolge und Anzahl der übergebenen Strings sollte mit der Reihenfolge und Anzahl der übergebenen Funktionen im Plot übereinstimmen. Mögliche Werte für `position` sind:

- -1: Rechts des Plots,
- 0: Matlab entscheidet (Standardeinstellung),
- 1: rechts oben,
- 2: links oben,

- 3: links unten,
- 4: rechts unten.

Es ist möglich (und oft sinnvoll), in Matlab-Strings LaTeX-Code zu verwenden. Für die Interpretation eines Strings als LaTeX-Code muss zunächst der `interpreter` von Matlab für den jeweiligen Textbereich umgestellt werden. Auf diesen kann man mittels der Setter-Funktion `set` zugreifen. Um an einer Legende (oder einem anderen Textbereich) mittels `set` Einstellungen vorzunehmen, muss diese vorab mit einer Variable initialisiert und somit handhabbar gemacht werden. Dies kann wie folgt geschehen:

```
l = legend('$f(x)$', 1);
set([l], 'interpreter', 'latex');
```

Zusätzlich kann man mittels `set` noch weitere Texteinstellungen vornehmen, wie bspw. die Veränderung der Textgröße über das Argument `'FontSize'` und durch ein Komma abgetrennt darauf folgend die gewünschte Schriftgröße.

Aufgabe 17 (Umgang mit Achsenbegrenzungen)
Stellen Sie die Funktion $f(x) = \frac{1}{(x-1)^2} + \frac{3}{(x-2)^2}$ in einem Plot dar. Überlassen Sie zunächst Matlab die Begrenzung der Achsen. Anschließend begrenzen Sie die y-Achse auf das Intervall $[0, 50]$.
Beschriften Sie die x- und y-Achse und fügen Sie eine Legende ein. In der Legende soll die Funktion $f(x)$ in LaTeX angegeben werden. Aktivieren Sie den LaTeX-Interpreter auch für die x- und y-Achse. Stellen Sie die Schriftgröße aller drei Textbereiche auf 15 ein. Als Auswertungsstellen betrachten Sie 300 gleichverteilte Stützstellen im Intervall $[0, 3]$.

Bemerkung: Um eine Funktion wie in Aufgabe 17 zu plotten, ist es nicht notwendig, diese als anonyme Funktion zu initialisieren. Erzeugen Sie stattdessen zunächst einen Stützstellenvektor x und anschließend einen Wertevektor y, der die einzelnen Auswertungen der Funktion an den Stützstellen von x enthält. Sie können y einfach mittels elementweiser Vektoroperationen auf x erstellen.

In Matlab ist es möglich, auch mehrere Plots in einem `figure`-Fenster anzeigen zu lassen. Der Befehl `subplot(m,n,p)` teilt das `figure`-Fenster in `m` Zeilen und `n` Spalten auf, womit dieses nun $m \cdot n$ Felder anbietet. `p` ist ebenfalls ein Skalar und beschreibt im `subplot` das mit `p` nummerierte Feld. Die Nummerierung erfolgt von links oben nach rechts unten. Für die bereits initialisierten (gleichdimensionalen) Vektoren `x1` und `y1` bzw. `x2` und `y2` erhalten wir das Beispiel:

```
subplot(2, 1, 1);
plot(x1, y1);
subplot(2, 1, 2);
loglog(x2, y2);
```

Auch eine unregelmäßige Aufteilung des `figure`-Feldes ist möglich. Hierzu interpretiert man `p` nicht als Skalar, sondern als Vektor.

Aufgabe 18 (Erzeugen von Subplots)
Schreiben Sie eine Skriptdatei, in der Sie die Funktionen x, x^2 sowie x^3 in einem `subplot` darstellen. Die Funktion x^3 soll dabei über mehrere `figure`-Felder reichen. Versehen Sie zudem jeden `subplot` mit einer Überschrift. Als Auswertungsstellen betrachten Sie 600 gleichverteilte Stützstellen im Intervall $[-3, 3]$.