Prof. Dr. Angela Kunoth
Max Glembock, Laslo Hunhold

Matlab Präsenzaufgaben — WS 2017/2018

Übungsblatt 3

Ausgabe: 25.10.2017

M-Dateien

Der Gebrauch von M-Dateien erweitert den Umfang von Matlab in vielerlei Hinsicht. M-Dateien sind das Pendant zu Programmen, Funktionen und Prozeduren in anderen Programmiersprachen. Fügt man eine Folge von Befehlen zu einer M-Datei zusammen, ergeben sich vielfältige Möglichkeiten, wie etwa

- an einem Algorithmus herum zu experimentieren, indem man eine Datei bearbeitet, anstatt eine lange Liste von Befehlen wieder und wieder zu tippen,
- einen dauerhaften Beleg für ein numerisches Experiment zu schaffen,
- nützliche Funktionen aufzubauen, die zu einem späteren Zeitpunkt erneut verwendet werden können,
- M-Dateien mit anderen Kollegen auszutauschen.

Technisch ist eine M-Datei eine Textdatei mit der Dateiendung `.m`. Man unterscheidet zwei Arten:

- **Skriptdateien:** Haben keine Ein- oder Ausgabeargumente und operieren auf Variablen im Workspace.
- **Funktionsdateien:** Enthalten eine `function`-Definitionszeile, akzeptieren Eingabeargumente und geben Ausgabeargumente zurück. Ihre internen Variablen sind lokal auf die Funktion beschränkt (sofern sie nicht als global deklariert wurden).

Sie können eine neue Skriptdatei bzw. eine neue Funktionsdatei in Matlab über den Reiter „New“ anlegen.

Ein Skript sammelt eine Folge von Befehlen, die wiederholt verwendet werden sollen oder in Zukunft noch gebraucht werden. Die ersten Zeilen der Skriptdatei sollten mit dem `%`-Symbol beginnen und Kommentare sein. Dies erlaubt das Einfügen von Erklärungen, die das Skript für andere Menschen zugänglicher machen. Das Skript zur folgenden Aufgabe 10 könnte z.B. so beginnen:

```
% -- randSin --
% Plottet die Sinusfunktion ausgewertet an n+2 zufälligen Stellen.
```

Angenommen das Skript wird auch unter dem Namen `randSin.m` gespeichert, dann ist die Eingabe von `randSin` in die Kommandozeile bzw. das Ausführen des Skripts gleichwertig zur Eingabe der einzelnen Zeilen des Skripts.

Aufgabe 10 (Approximation der Sinus-Funktion)

Erstellen Sie eine neue Skriptdatei `randSin`, die die Sinus-Funktion an $n = 10$ zufälligen Stellen im Intervall $[0, 2\pi]$ sowie an 0 und 2π selbst auswertet und in einem Plot darstellt. Generieren Sie hierfür zunächst 10 Zufallszahlen in einem Vektor `x` und ordnen Sie diese der Größe nach aufsteigend mit dem Befehl `sort(x)`. Speichern Sie den elementweisen Sinus von `x` in einem Vektor `y`. Plotten Sie die Funktionswerte `y` an den Auswertungsstellen von `x` mit dem Befehl `plot(x,y)` in Polygonzugdarstellung.
Warum ist es wichtig, dass die Auswertungsstellen in `x` zuvor geordnet werden?

Wichtig: Achten Sie bei der Ausführung des Skripts darauf, dass unter Current Folder der Ordner ausgewählt ist, in dem Sie das Skript gespeichert haben.

Funktionsdateien werden in Matlab auf die gleiche Weise verwendet wie die bereits existierenden Matlab-Funktionen (`sin`, `eye`, ...). Wir möchten das Anlegen einer Funktionsdatei an einem Beispiel verdeutlichen:

```
function y = maxentry(A)
% MAXENTRY
% Berechnet das größte absolute Element einer Matrix A.
y = max(abs(A(:));
```

Die erste Zeile fängt mit dem Schlüsselwort `function` an, gefolgt vom Ausgabeargument `y` und dem Gleichheitszeichen. Rechts von = kommt der Funktionsname `maxentry`, gefolgt vom Eingabeargument `A` in Klammern (im Allgemeinen kann es beliebig viele Ein- und Ausgabeargumente geben). Der Funktionsname

muss der gleiche sein wie der Name der M-Datei, die die Funktion enthält.
Die zweite Zeile der Funktionsdatei heißt H1-Zeile (help 1). Sie sollte eine Kommentarzeile sein. Alle Kommandozeilen – beginnend mit der ersten bis zur ersten Nichtkommentarzeile (in der Regel eine Leerzeile, um die Lesbarkeit des Codes zu erhöhen) – werden beim Aufruf von `help function_name` angezeigt. Daher sollten diese Zeilen die Funktion und ihre Argumente beschreiben. Alternativ können die Kommentare auch oberhalb der `function`-Zeile angegeben werden (siehe vorheriges Beispiel). Funktionsdateien können (analog zu Skripten) über das Command Window aufgerufen werden.
Der Befehl `help` funktioniert ebenfalls mit Skriptdateien; er zeigt die Anfangsfolge an Kommentaren an.

Aufgabe 11 (Approximation der Sinus-Funktion – 2)
Wandeln Sie die Skriptdatei von Aufgabe 10 in eine Funktionsdatei namens `randSinFunc` um, die als Eingabeparameter die Anzahl der auszuwertenden Stellen n hat. Lassen Sie sich zusätzlich das Maximum der Auswertungen ausgeben.

Flusskontrolle und Operatoren

Matlab hat vier Strukturen für die Flusskontrolle: die `if`-Abfrage, die `for`-Schleife, die `while`-Schleife und den `switch`-Befehl. Die einfachste Form der `if`-Abfrage lautet:

```
if expression
  statements
end
```

Bei einer `if`-Abfrage werden die `statements` dann ausgeführt, wenn die `expression` wahr ist. Wie in anderen Programmiersprachen auch, können in Matlab für die Darstellung der `expression` relationale Operatoren verwendet werden. Die relationalen Operatoren von Matlab sind:

==	gleich
~=	ungleich
<	weniger als
>	größer als
<=	kleiner oder gleich
>=	größer oder gleich

Beachten Sie, dass in Matlab ein einzelnes Gleichheitszeichen = eine Zuweisung angibt und **nie** auf Gleichheit testet. Vergleiche zwischen Skalaren mit den obigen relationalen Operatoren erzeugen eine logische 1, wenn die Relation wahr ist, und eine logische 0, wenn sie falsch ist. Folglich gibt die Eingabe von `3==4` eine `0` und die Eingabe von `3~=4` eine `1` zurück.
Man kann mehrere relationale Operatoren in Matlab mittels logischer Operatoren verknüpfen. Die logischen Operatoren von Matlab sind:

&&	logisches und
||	logisches oder
~	logisches nicht
`all`	wahr, wenn alle Elemente eines Vektors von Null verschiedenen sind
`any`	wahr, wenn wenigstens ein Element eines Vektors von Null verschiedenen ist

Befehle, die nur ausgeführt werden sollen, wenn `expression` falsch ist, können nach `else` eingefügt werden:

```
if expression
  statements1
else
  statements2
end
```

Schließlich kann mit `elseif` auch noch ein weiterer Test hinzugefügt werden (beachte, dass zwischen `else` und `if` kein Leerzeichen stehen darf).

Die `for`-Schleife ist einerseits eine der wichtigsten Strukturen in Matlab, andererseits vermeiden viele Programmierer, die kurzen und schnellen Code produzieren wollen, dessen Verwendung. Die Syntax lautet:

```
for variable = ausdruck
  statements
end
```

Üblicherweise ist `ausdruck` ein Vektor (bspw. über den Doppelpunkt-Operator oder die Klammernotation definiert). Die Befehle werden für jedes Element von `ausdruck` ausgeführt, wobei `variable` dem entsprechenden Element von `ausdruck` zugeordnet ist. Mehrere `for`-Schleifen können verschachtelt werden. In

diesem Fall hilft es, die verschachteteln Schleifen einzurücken, um die Lesbarkeit des Textes zu erhöhen.

Die `while`-Schleife von Matlab hat die Form:

```
while ausdruck
  statements
end
```

Die `statements` der `while`-Schleife werden ausgeführt, solange `ausdruck` wahr ist. So sucht das folgende Beispiel mittels einer `while`-Schleife die größte in Matlab darstellbare Zweierpotenz `max_2`:

```
tmp = 1;
while(tmp < inf)
  max_2 = tmp;
  tmp = max_2*2;
end
```

Sowohl eine `while`-Schleife als auch eine `for`-Schleife können mit dem Befehl `break` beendet werden. Bei verschachtelten `for`-Schleifen gelangt man mit `break` wieder auf die nächsthöhere Ebene zurück.

Aufgabe 12 (Umgang mit Flusskontrollen)

a) Schreiben Sie mittels zwei verschachtelten `for`-Schleifen und einer `if`-Anweisung eine Funktion `maxentry`, die das größte absolute Element einer Matrix **beliebiger Größe** wiedergibt.

b) Schreiben Sie eine rekursiv definierte Funktion `factorial`, die die Fakultät einer natürlichen Zahl n berechnet. Die Funktion soll vorher prüfen, ob es sich bei n um eine natürliche Zahl handelt. Hilfreich sind hierbei die Funktionen `floor(n)` und `ceil(n)`, die n ab- bzw. aufrunden. Sollte n keine natürliche Zahl sein, so soll 0 wiedergegeben werden.

c) Schreiben Sie ein Skript `xmin`, welches in einer `while`-Schleife die kleinste von Null verschiedene Gleitkommazahl approximiert, die man in Matlab berechnen kann.

Wichtig:

- Sollte sich Ihr Programmcode in einer Endlos-Schleife aufhängen, so können Sie die aktuelle Matlab-Berechnung durch die Tastenkombination `Strg+C` abbrechen.
- Sie können Ihren Matlab-Code automatisch einrücken und ordnen lassen. Markieren Sie dazu die gewünschten Stellen und drücken Sie die Tastenkombination `Strg+I`.

Der Kontrollbefehl `switch` besteht aus einer Kopfzeile `switch ausdruck`, gefolgt von einer Liste von `case ausdruck befehl`, die optional mit `otherwise ausdruck` enden und mit einem `end` beendet werden. Mehrere Cases lassen sich mit geschweiften Klammern `{...}` (der sogenannten Cell-Array-Notation) zusammenfassen. Ein Beispiel gibt die folgende `switch`-Abfrage, die prüft, ob es sich bei einer Zahl `p` um eine Primzahl zwischen 1 und 9 handelt:

```
switch p
  case 1
    disp('p ist die 1')
  case {2,3,5,7}
    disp('p ist prim')
  case {4,6,8,9}
    disp('p ist nicht prim')
  otherwise
    disp('p liegt nicht zwischen 1 und 9')
end
```

Wichtig: `switch`-Cases akzeptieren **nur** Cell-Arrays oder Strings als annehmbare Werte. Die Eingabe von Vektoren gibt zwar keine Fehlermeldung zurück, doch wird diese von Matlab auch nicht angenommen.

Aufgabe 13 (Umgang mit Flusskontrollen – 2)

Schreiben Sie eine Funktion `norm_own(x,p)`, die zu einem Eingabevektor $x \in \mathbb{R}^n$ und zu einer Zahl $p \in \{1, 2, \infty\}$ die entsprechende p-Norm berechnet. In Matlab wird ∞ über den Befehl `inf` dargestellt. Für die Normen gilt:

- $\|x\|_1 = \sum_{i=1}^{n} |x_i|$,
- $\|x\|_2 = \left(\sum_{i=1}^{n} x_i^2\right)^{1/2} = (x^T x)^{1/2}$,
- $\|x\|_\infty = \max_{1 \le i \le n} |x_i|$.

Falls $p \notin \{1, 2, \infty\}$ soll über den Befehl `error(String)` eine Fehlermeldung ausgegeben werden. Macht es einen Unterschied, ob x als Zeilen- oder Spaltenvektor eingegeben wird?