Wima 1 - Praktikum (Woche 2)

Lernziele

In diesem Praktikum sollen Sie üben und lernen:

- Arbeiten mit Matrizen und Vektoren
- Schreiben einfacher Skripte in MATLAB
- Erstellen eigener Funktionen in MATLAB
- Kennenlernen der `for`-Anweisung

Am Anfang wollen wir Ihnen einen kurzen Überblick über Skripte und Funktionen in MATLAB geben.

Beantworten Sie danach bitte erst einige Fragen bzw. einige off-line Aufgaben, bevor bevor Sie sich einloggen!

Skripte und Funktionen in Matlab

Skripte

Es ist sehr unübersichtlich, alle Berechnungen im Command Window durchzuführen. Matlab kann Dateien (so genannte M-Files) erstellen, in denen man seine Befehle speichern kann. Über den Menüpunkt File $\mapsto$ New $\mapsto$ M-File oder durch Eingabe von `edit` kann am MATLAB-Prompt eine neue Datei geöffnet werden. Die Befehle werden nun in diese Datei geschrieben, die Datei unter einem Namen abgespeichert. Sie erhält die Matlab-Endung `.m`. Der Aufruf der Datei geschieht im Kommandofenster durch Aufruf des Dateinamens. Die Endung `.m` muss dabei nicht angefügt werden. Im Folgenden wird ein einfaches Programm besprochen. Zunächst wird der gesamte Quellcode angegeben, anschließend werden die einzelnen Befehlszeilen erläutert. Das Programm `ex02x01.m`, welches zu einer gegebenen Zahl k die Funktion $x \cdot \sin(k/x)$ graphisch darstellt, könnte z.B. so aussehen:

```
% ex02x01.m
% Dieses Programm plottet zu gegebenem k
% die Funktion x * sin(k/x)

disp('Dieses Programm liest eine Zahl k ein ')
disp('und stellt die Funktion x * sin(k/x) graphisch dar.');
disp(' ');
k = input('Bitte nun eine Zahl k eingeben: ');
disp(' ');
disp(['Die eingegebene Zahl war k = ',num2str(k)]);

x = 0.001:1e-5:0.3;
y = x.*sin(k./x);
plot(x,y)
grid on
title('Graph von x*sin(k/x)')
xlabel('x-Werte')
ylabel('Funktionswerte')

hold on
plot(x,x,'k:',x,-x,'k:')
axis([0,0.3,-0.3,0.3])
hold off
```

Die resultierende Ausgabe zu obigem m-File und eingegebenem Wert $k = 5$ ist in der folgenden Grafik dargestellt

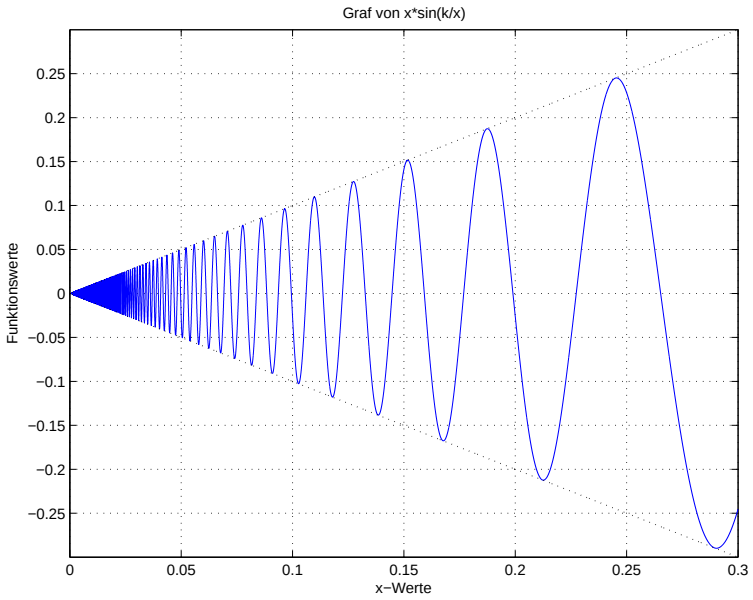

Abbildung 1: Darstellung der Funktion $x \sin(k/x)$ im Intervall $[0.003, 3]$ mit Achsenbeschriftung und Gitternetzlinien mit Hilfe des m-Files `ex02x01`

Kommentare werden mit % versehen. Alles, was in einer Zeile nach einem % folgt, ist ein Kommentar. Es ist wichtig, jedes Programm zu kommentieren. Speichert man das oben geschriebene Programm unter dem (noch nicht in der Matlab-Funktionen-Datenbank vorhandenen) Titel `ex02x01.m`, so gibt der Befehl

```
≫ help ex02x01
```

den Kommentar

```
ex02x01.m
Dieses Programm plottet zu gegebenem k
die Funktion x * sin(k/x)7
```

aus. Es empfiehlt sich, zu jedem Programm, das man schreibt, eine solche Erläuterung zu schreiben und wirklich jeden Schritt im Programm zu kommentieren. Wenn die ersten Zeilen eines m-Files aus einem Kommentar bestehen, gibt der Befehl help `Dateiname.m` diese Zeilen aus. Alle in MATLAB enthaltenen Funktionen haben eine kurze Erläuterung als Kommentar in den ersten Zeilen stehen.

Textausgaben mit dem Text ... im Kommandofenster werden mit Hilfe des Befehls `disp(' ... ')` erreicht. `disp(' ')` erzeugt eine Leerzeile, was das Programm manchmal übersichtlicher macht. Das Programm kann auch Zahlen und Berechnungen des Programms im Fließtext ausgeben, in dem man den Befehl `disp([' '])` wie oben angewendet in Kombination mit `num2str()` benutzt:

```
disp(['Die von Ihnen eingegebene Zahl war k = ',num2str(k)]);
```

Der Befehl `num2str` verwandelt eine Zahl oder eine Matrix in eine Zeichenkette, die ausgegeben werden kann. Parameter können entweder direkt im m-File definiert werden, oder wie oben mittels `input(' ')` eingelesen werden:

```
k = input('Bitte nun eine Zahl k eingeben: ');
```

Der Text `Bitte nun eine Zahl k eingeben:` erscheint nach dem Prompt, der Variablen k wird der Wert der eingegebenen Zahl zugewiesen.
Dadurch, dass nach einigen Anweisungen ein Semikolon steht, wird die Ausgabe (des Ergebnisses der Berechnung) im Kommandofenster unterdrückt.

Funktionen

Oft ist es sinnvoll, eigene Funktionen zu programmieren, die dann in einem Programmdurchlauf ausgeführt werden. Es gibt dann ein Hauptprogramm, in welchem mehrere Funktionen aufgerufen werden. Funktionen werden genauso wie gewöhnliche m-Files geschrieben. Der einzige Unterschied besteht darin, dass das erste Wort `function` sein muss (abgesehen von Kommentarzeilen). In der ersten Zeile wird dann der Name der Funktion definiert (relevant ist bei älteren MATLAB-Versionen der Name der Datei) und es werden die Variablen eingelesen. Als Beispiel soll nun ein kleines Programm dienen, welches zwei Werte a und b einliest und dann Berechnungen zu dem Rechteck durchführt. Dazu schreiben wir zunächst die Funktion `FlaecheninhaltRechteck.m`.

```
1 function A = FlaecheninhaltRechteck(a,b)
2 % FlaecheninhaltRechteck(a,b) berechnet den Flaecheninhalt
3 % des Rechtecks mit den Seiten a und b
4 A = a * b;
```

Die Variablen unmittelbar hinter `function`, hier also `A`, bezeichnen die Werte, die berechnet und ausgegeben werden. Die Variablen hinter dem Funktionsnamen in den runden Klammern bezeichnen die Werte, die eingelesen werden.

Man beachte, dass MATLAB im Allgemeinen keine Zwischenräume benötigt, um Befehle eindeutig zu verstehen.

Weiterhin soll die Diagonale des Rechtecks mit der Funktion `DiagonaleRechteck.m` berechnet werden.

```
1 function d = DiagonaleRechteck(a,b)
2 % DiagonaleRechteck(a,b) berechnet die Diagonale des
3 % Rechtecks mit den Seiten a und b
4 % mit Hilfe des Satzes von Pythagoras.
5 d = sqrt(a^2 + b^2);
```

Das Haupt-Programm Rechteck.m könnte nun so aussehen:

```
1 % Das Programm Rechteck.m liest zwei Werte a und b ein
2 % und berechnet den Flaecheninhalt und die Diagonale
3 % des Rechtecks mit Kantenlängen a,b
4 clear all
5 disp('Programm liest Kantenlaengen a, b ein und berechnet')
6 disp('den Flaecheninhalt und die Diagonale des Rechtecks')
7 disp(' ')
8 a = input('Bitte a eingeben: ');
9 b = input('Bitte b eingeben: ');
10 A = FlaecheninhaltRechteck(a,b);
11 d = DiagonaleRechteck(a,b);
12 disp(['Der Flaecheninhalt des Rechtecks ist A = ',num2str(A)])
13 disp(['und die Diagonale betraegt d = ',num2str(d)])
```

Das `clear all` in der 4. Zeile löscht alle Variablen im Workspace, d.h. der Speicher ist wieder im „Startzustand“. Bei so kleinen Programmen erscheint es noch nicht wirklich sinnvoll, Unterroutinen als Funktionen zu schreiben. Dies ist aber bei komplizierteren Programmen und besonders wenn eine Routine häufig benutzt werden muss, sehr hilfreich.

Sollen mehrere Werte innerhalb einer Funktion berechnet werden, schreibt man

```
function [A, B, C] = FunktionsName(a,b,c,d,e);
```

Die Variablen hinter dem Funktionsnamen in den runden Klammern bezeichnen die Werte, die eingelesen werden. In den eckigen Klammern nach dem Wort `function` stehen die Variablen, die von der Funktion nach dem Funktionsdurchlauf wieder zurückgegeben werden. Also die Werte, die in der Funktion berechnet werden sollen.

Vorsicht bei der Vergabe von Funktionsnamen! Sehr viele Fehler entstehen durch eine doppelte Vergabe eines Funktionsnamens. Ob der von mir gewählte Name bereits vergeben ist, kann ich mit Hilfe der Funktion `exist` überprüfen.

Ausgabe	Bedeutung
0	Name existiert noch nicht
1	Name ist bereits für eine Variable im Workspace vergeben
2	Ist ein bereits bestehendes m-File oder eine Datei unbekannten Typs
3	Es existiert ein mex-File mit diesem Namen
4	Es existiert ein MDL-File mit diesem Namen
5	Name ist an eine Matlab Funktion vergeben (z.B. sin)
6	Es existiert ein p-File mit diesem Namen
7	Es existiert ein Verzeichnis mit diesem Namen

```
≫ exist d
ans =
     1
≫ exist tan
ans =
     5
```

Die `for`-Schleife

Die `for`-Schleife eignet sich, wenn Berechnungen wiederholt durchgeführt werden. Die `for`-Schleife zur n-fachen Ausführung einer Befehlssequenz besitzt die folgende Syntax:

```
for Variable = Matrix/Cell/Feld
     Befehle
end
```

Der Variablen werden nacheinander die Spalten der Matrix bzw. der Cell/Feld zugewiesen. Im Falle mehrdimensionaler Felder werden analog die Spalten aller Teilmatrizen durchlaufen. Für jede Spalte werden die Befehle einmal ausgeführt. Ein n-facher Schleifendurchlauf kann mittels

```
for zaehler = 1 : n
```

realisiert werden. Ein vorzeitiger Abbruch der Schleife ist durch Angabe des Befehls `break` möglich (z.B. innerhalb einer `if`-Abfrage). Sei als Beispiel eine (n, n)-Matrix A mit den Einträgen $a_{j,k} = 1/(j + k - 1)$ für $j, k = 1, \ldots, n$ initialisiert. Mit Hilfe der `for`-Schleife kann man folgendes Programm `ForSchleife.m` schreiben.

```
% ForSchleife.m dient zum Lernen der for Schleife
% und berechnet eine Matrix.
clear all
n = input('Bitte eine Zahl n eingeben: ');
% Initialisieren der Matrix mit Nullen
A= zeros(n);
% Aufbau der Matrix:
for j=1:n
  for k=1:n
    A(j,k)=1/(j+k-1);
  end
end
```

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.	`exist`	a.	Zugriff auf die MATLAB-Dokumentation.
_____	2.	`function`	b.	Leitet eine Kontrollanweisung ein.
_____	3.	`plot`	c.	Ist äquivalent zu `[5,4,3,2,1]`.
_____	4.	Fortran	d.	Benötigt zur Ausführung einen Compiler.
_____	5.	`helpdesk`	e.	Standardvariablenname für Ergebnisse.
_____	6.	`for`	f.	Leitet die Definition einer Funktion ein.
_____	7.	`Inf`	g.	Ist eine Grafikfunktion .
_____	8.	`[5:-1:0.99]`	h.	Liefert Aussage über die Vergabe von Funktionsnamen.
_____	9.	`ans`	i.	Entspricht dem (symbolischen) Wert ∞.
_____	10.	`end`	j.	Beendet eine Kontrollanweisung.

Ihre Antwort:

Füllen Sie die Lücken aus

Ergänzen Sie die folgenden Sätze.

11. Algorithmen bezeichnet man als _____, wenn sie Funktionen enthalten, die sich direkt oder indirekt selbst aufrufen .
12. Durch die Entwicklung eigener MATLAB- _____ ist es möglich, die Basisfunktionalität von MATLAB zu erweitern.
13. Mit dem MATLAB-Befehl _____ rundet man auf die nächst kleinere ganze Zahl .
14. Das Wort _____ kommt in jeder MATLAB-Funktion vor.
15. Alle Datenobjekte eines Skripts sind _____ für den aufrufenden Programmteil.
16. Zugriff auf die MATLAB-Dokumentation liefern die Befehle _____ .
17. Matrizen werden in MATLAB _____-weise gespeichert.
18. Der Befehl `length(vek)` liefert die _____ des Vektors `vek` .
19. Mit dem _____ kann man das Laufzeitverhalten eines MATLAB-Programms analysieren.
20. Matrizen A, B mit der gleichen Dimension können mit dem Befehl `.^` _____ 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. Wie lautet die Ausgabe des folgenden Funktion, wenn Sie `peter(eye(2))` ausführen?

```
1 function value = peter(A)
2 value = multipliziere(A);
3
4 function B = multipliziere(A)
5 B = 3*A;
6 B = B-1;
```

Ihre Antwort:

22. Wie und warum lautet die Ausgabe des folgenden Skripts?

```
1 clear all
2 2^3^4
```

Ihre Antwort:

23. Wie lauten die Werte von `b` und `c` nach Ausführen des folgenden Skripts?

```
1 clear all
2 a = pi;
3 b = ceil(a);
4 c = floor(a);
```

Ihre Antwort:

24. Wie lautet der Wert von r, nach dem die Zeilen 1-3 ausgeführt wurden?

```
1 A = [1 2 3; 4 5 6];
2 i = size(A(2:end,2:end),2);
3 r = i^2;
```

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 jeden dafür vorgesehenen Bereich unter der Problemstellung. Falls das Segment keinen Fehler enthält, schreiben Sie einfach „kein Fehler“. [Bemerkung: Es kann sein, dass ein Programm mehrere Fehler enthält.]

25. Die folgende Matlab-Funktion soll das Produkt der beiden Matrizen A und B zurückgeben.

```
1 function C=MatrixProdukt(A,B);
2   for i = 1:size(A,1)
3     for j = 1:size(B,2)
4       C(i,j) = C(i,j) + A(i,:) .* B(:,j);
5     end
6   end
7 end
```

Ihre Antwort:

26. Das folgende Skript sollte die Funktion $y(x) = \arctan(x^2)$ auf dem Bildschirm ausgeben:

```
1 clear all;
2 x = linspace(-1,1,100);
3 fx = arctan(x^2);
4 plot(x,fx,'k-');
```

Ihre Antwort:

Praktikumsaufgabe - Einfacher Funktionsaufruf

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.

27. Schreiben Sie ein Skript `summe.m`, welches die Summe der ersten 10 natürlichen Zahlen mit Hilfe einer `for`-Schleife berechnet

28. Schreiben Sie eine Funktion `produkt.m`, welche das Produkt der ersten n natürlichen Zahlen mit Hilfe einer `for`-Schleife als Rückgabewert liefert.

29. Nach Archimedes lässt sich π durch folgende Vorgehensweise ermitteln: Man berechnet den halben Umfang u_n eines regelmäßigen 2^n-Ecks, das dem Einheitskreis eingeschrieben ist, und führt dies für wachsendes n durch. Wegen $\lim_{n\to\infty} u_n = \pi$ erhält man das folgende numerische Verfahren zur näherungsweisen Berechnung von π:

$$\begin{aligned} u_1 &= 2 \\ u_n &= 2^n \sqrt{\frac{1}{2}\left(1 - \sqrt{1-(u_{n-1}/2^{n-1})^2}\right)} \quad \text{für } n = 2,3,4,\ldots,n_{\max} \end{aligned}$$

Schreiben Sie eine Funktion `approxpi.m`, welche zum Eingabeparameter $n_{\max}$ den Vektor $u = (u_1, u_2, \ldots, u_{n_{\max}})$ als Rückgabe liefert.

Wenn Sie nun folgendes Skript ausführen

```
1 % plotte relativen Fehler bei der Approximation von pi
2 clear all
3 p = approxpi(30);
4 relf = abs((p-pi)/pi);
5 semilogy(1:30,relf,'*-')
```

erhalten Sie folgende Darstellung.

Bemerkung: Mathematisch gilt zwar $\lim_{n\to\infty} u_n = \pi$, bei der numerischen Realisierung des Verfahrens treten jedoch Auslöschungseffekte auf, so dass die numerisch erhaltenen Näherungswerte $\tilde{u}_n$ nicht gegen π konvergieren. Dies ist daraus zu erklären, dass in der innersten Wurzel der Wert $u_{n-1}/2^{n-1}$ schnell gegen Null konvergiert und schon ab relativ kleinen Werten n nicht mehr als Gleitpuntzahl darstellbar ist. Dadurch wird die innere Wurzel 1 und die äußere Wurzel Null.

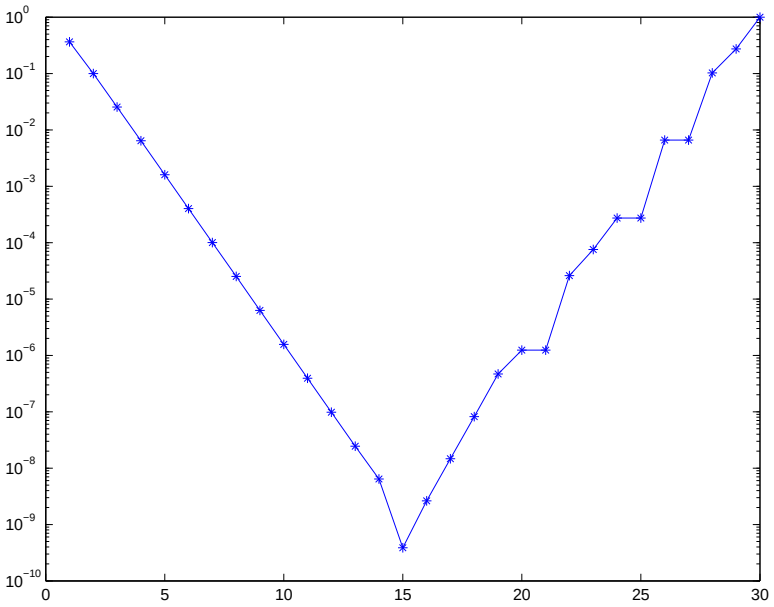

Abbildung 2: Relativer Fehler bei der Approximation von π.