Lernziele

In diesem Praktikum sollen Sie üben und lernen:

- Umgang mit der Matlab-Umgebung
- Schreiben einfacher Skripte und Funktionen in Matlab
- Berechnung der Komplexität von Algorithmen
- Grafische Darstellung der Ergebnisse von Laufzeitmessungen

Am Anfang wollen wir Ihre Kenntnisse über das Lösen von linearen Gleichungssystemen und die Programmierung in Matlab etwas auffrischen. Dazu stellen wir Ihnen einige Fragen bzw. einige off-line Aufgaben.

Machen Sie bitte zuerst die folgenden off-line Übungen, bevor Sie sich einloggen!

Off-line Aktivitäten

Kurz und knapp

Geben Sie bitte eine kurze Antwort zu jeder der folgenden. Ihre Antwort sollte so kurz und präzise wie möglich sein; versuchen Sie es mit zwei bis drei Sätzen.

1. Worin unterscheidet sich in Matlab eine Funktion von einem Skript?
   Ihre Antwort:

2. Wozu verwendet man am Anfang eines Skripts häufig die `clear all` Anweisung?
   Ihre Antwort:

3. Was besagt die Fehlermeldung **`Index exceeds matrix dimensions`**?
   Ihre Antwort (geben Sie ein Beispiel an):

4. Was ist das Ergebnis von **`a=0/0;`** in Matlab? Und wie fährt das Programm nach Auswertung dieser Anweisung fort?
   Ihre Antwort:

5. Welche Werte nehmen **`k`** und **`m`** in Matlab in den folgenden for-Schleifen an?

```
for k = 1:2:3
   ...
end

for m = [1, 4, 9]
   ...
end
```

   Ihre Antwort:

6. Welcher Unterschied besteht in Matlab in den Anweisungen
`y = a/b;` und `y = b\a;`?
Ihre Antwort:

7. Welche Ergebnisse liefern die Funktionen **`why, whos, who, clc, format compact`** in Matlab?
Ihre Antwort:

8. Welche Zeichen darf man zur Definition einer Variablen in Matlab verwenden?
Ihre Antwort:

Programmausgaben

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

9. Wie lautet die Ausgabe des folgenden Programms?

```
1        A = [2,3,4;3 4 5];
2        B = 2*A;
3        B(1,2) = 3;
4        B
```

Ihre Antwort:

10. Wie lautet der Wert der Variablen **a,b,c,** nachdem die folgenden Anweisungen ausgeführt wurden?

```
1        a = 8\4;
2        b = [1:3].^2 * 9 / 3 + 3 - ( 7 + 3 ) / 5;
3        c = 12 * ( 6 \ 4 ) + 3 - 7 + 9 / 3;
```

Ihre Antwort:

11. Vorausgesetzt der Benutzer gibt 3 ein, wie lautet die Ausgabe des folgenden Programms?

```
1        clear all
2        number = input('Geben Sie eine natürliche Zahl ein!');
3        tmp = 0;
4        for k = -1 : number
5          tmp = tmp + k;
6        end
7        str = sprintf('Das Ergebnis lautet %g', tmp);
8        disp(str)
```

Ihre Antwort:

12. Welche Dimension hat **`A`**, nachdem die Zeilen 1-6 ausgeführt wurden?

```
1        clear all
2        A = rand(4);
3        B = zeros(4,2);
4        C = A^3*B;
5        C(1,:)=[];
6        A = C(:,2:end);
```

Ihre Antwort:

13. Vorausgesetzt der Benutzer übergibt 3 an die Funktion, wie lautet der Rückgabewert?

```
1        function value = fakultaet(n)
2        % berechnet die Fakultaet von n, d.h. n!
3        value = 1;
4        for k = 0 : n
5          value = value * k;
6        end
```

Ihre Antwort:

14. Welchen Wert besitzt `i,` nachdem die Zeilen 1-4 ausgeführt wurden?

```
1        i = 324;
2        while i
3          i = i-1;
4        end
5        i
```

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 logischen 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.]

15. Die folgende Matlab-Funktion soll das Produkt der ersten **n** natürlichen Zahlen zurückgeben.

```
1        function tmp = multipliziere(n);
2        tmp = 0;
3        for k = 1 : n
4          tmp = tmp * k;
5        end
```

Ihre Antwort:

16. Das folgende Programm sollte **Hello world!** auf dem Bildschirm ausgeben:

```
1        str = 'Hello world!;
2        disp(Str)
```

Ihre Antwort:

17. Das folgende Programm sollte das Skalarprodukt der beiden Spaltenvektoren **x, y** auf dem Bildschirm ausgeben:

```
1        x = [1; 4; 3];
2        y = [3; 2; 7];
3        z = x*y';
2        disp(sprintf('Das Skalarprodukt von x und y lautet %f',z));
```

Ihre Antwort:

18. Das folgende Programm sollte die zweite Zeile der Matrix **A** auf dem Bildschirm ausgeben:

```
1        A = rand(3);
2        z = A(:,2);
3        z
```

Ihre Antwort:

Praktikumsaufgabe 1 – 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.

• Problembeschreibung
Rufen Sie die folgende Funktion im Kommando-Fenster von Matlab auf.

```
% ex01x01.m
% simple program using transpose and array power

function A = ex01x01          %% hand over the input parameter n
k = [1,2,3,4];                %% create a vector with entries 1,...,n
k = k';
A = [k.^1,k.^2,k.^3,k.^4];    %% create the matrix [k.^1,k.^2,...,k.^n]
```

Sie sollten auf dem Bildschirm eine 4 x 4 - Matrix ausgegeben bekommen mit Einträgen $a_{jk} = j^k$.

Modifizieren Sie `ex01x01` so, dass Sie eine Funktion `ex01x02` mit den folgenden Eigenschaften erhalten:

- Die Funktion bekommt die Dimension n als Parameter übergeben.
- Als Rückgabewert erhält man eine n x n – Matrix mit mit Einträgen $a_{jk} = j^k$

• Vorlage
Die Datei `ex01x01.m` ist eine lauffähige Matlab-Funktion. Die Datei können Sie sich auf der Vorlesungshomepage downloaden.

Praktikumsaufgabe 2 – Einfache Matrix-Vektor-Operationen

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.

• Problembeschreibung

Rufen Sie die folgende Funktion im Kommando-Fenster von Matlab auf.

```
% ex01x03.m
% compute solution of A * x = b with lower triangular matrix A

function x = ex01x03(A,b)
if prod(diag(A)) == 0
  error('Matrix is not regular!')
end
n = size(A,1);
x = zeros(n,1);
for j = 1:n
  for k=1:j-1
    b(j) = b(j) - A(j,k) * x(k);
  end
  x(j) = b(j) / A(j,j);
end

% A = tril(rand(5));  % create random lower triangular matrix
% x = rand(5,1);      % create random vector x
% b = A*x;            % create corresponding rhs b = A * x
% [ex01x03(A,b),x]    % compare computed and exact solution
```

Sie sollten, wenn Sie die oben (ausdokumentierten) genannten Anweisungen nacheinander eingeben, zwei gleiche Vektoren erhalten, bzw. eine 5x2-Matrix.

Modifizieren Sie `ex01x03` so, dass Sie eine Funktion `ex01x04` mit den folgenden Eigenschaften erhalten:

- Die Funktion berechnet die Lösung von A x = b für obere Dreicksmatrizen.
- Als Rückgabewert erhält man den Spaltenvektor x.

• Ergebnisverifikation

Schreiben Sie ein kurzes `testit.m` Skript mit den Anweisungen

```
rand('seed',0);         % reset random number generator
A = triu(rand(5));      % create random upper triangular matrix
x = rand(5,1);          % create random vector x
b = A*x;                % create corresponding rhs b = A * x
y = ex01x04(A,b)
```

Vergleichen Sie x und y!

Lab Exercise 3 – Random Walk

Read the problem description; then study the template code. Replace the **%%** comments within the template with Matlab statements. Run the program.

• Description of the Problem

Simulate the walk of a drunkard in a square street grid. One might expect that on average the person might not get anywhere because the moves in different directions cancel each other out in the long run, but in fact it can be shown that the person eventually moves outside any finite region.

a.) Execute the following Matlab script and find out how it works.

```
% ex01x05.m
% script creates and plots a random walk
clear all

A = zeros(20,20);           % declare grid of dimension 20x20
                            % (initialized with zeroes)

x = 10; y = 10;             % define start cell
A(x,y) = 1;                 % mark start cell with '1'

seed = input('Enter a seed (integer): ');
                            % to produce a different sequence,
rand('state',seed);         % invoke rand with different 'seeds'


for i=1:9
  r = rand;                 % pick randomly a direction
  if r < 0.25
    x = x+1;                % east
  elseif r < 0.5
    y = y+1;                % north
  elseif r < 0.75
    x = x-1;                % west
  else
    y = y-1;                % south
  end
  A(x,y)= 1;                % mark new block

end
spy(A);
```

b.) In `ex01x05.m` the drunkard makes only 9 steps. Hence, he will always move inside the grid. Write a program `Ex01x06.m` (by modifying `Ex01x05.m`) by adding some `if`-statement which excludes 'marking' if the actual `x` or `y` are not in the range from 1 to 20. Nevertheless, `x` or `y` should be increased respectively decreased.

• Template

The program template represents a subset of `ex01x05.m`, with one or more key lines of code replaced with comments.

```
      y = y-1;                % south
```

```
    end

    if %% add a condition which insures that x and y
       %% are both >=1 and <=20
      A(x,y) = 1;                  % mark new block
    end
  end
  spy(A);
              // plot grid
```

c.) Write a program `ex01x07.m` (by modifying `ex01x05.m`) which asks the user continuously to plot another path until the answer is **`N`** or **`n`**.

• Template

The program template represents a subset of `Ex01x05.m`, with one or more key lines of code replaced with comments.

```
% ex01x07.m
% function creates and plots a random walk
clear all

flag = 1;
while flag
  A = zeros(20,20);            % declare grid of dimension 20x20
                               % (initialized with zeroes)

  x = 10; y = 10;              % define start cell
  A(x,y) = 1;                  % mark start cell with '1'

  seed = input('Enter a seed (integer): ');
                               % to produce a different sequence,
  rand('state',seed);          % invoke rand with different 'seeds'


  for i=1:9
    ...                        % pick randomly a direction
                               % east, north, west, south
    A(x,y)= 1;                 % mark new block
  end
  spy(A);

  reply = input('Do you want more? Y/N [Y]: ','s');
  if reply =='n' | reply == 'N'
    flag = 0;
  end
end
```