Matlab Tutorials

Das *LiveScript* zu dem Tutorial 5.

Lösung: Aufgabe aus Tutorial 4

```
iIn  = 12;          % integer/double
bIn  = true;        % bool
szIn = 'Loesung';   % string
caIn = {iIn, bIn, szIn};    % cell array
hIn  = @(x) (x - 1).^2;     % handle
tIn  = datetime();  % datetime object

stOut = tut4_function(iIn, bIn, szIn, caIn, hIn, tIn);

disp(stOut)
```

Funktion mit dem Namen "tut4_function.m":

```
function [stOut] = tut4_function(iIn, bIn, szIn, caIn, hIn, tIn)
% TUT4_FUNCTION adds serveral different data types to a struct!
%
% TUT4_FUNCTION contains some very usefull calculations from the
% "Matlab Tutorials".

% Author: Julian Kahnert (c) IHA @ Jade Hochschule

stOut = struct();
stOut.integer   = iIn;
stOut.bool      = bIn;
stOut.string    = szIn;
stOut.cellarray = caIn;
stOut.handle    = hIn;
stOut.date      = tIn;
end
```

Tutorial 5.1

- Preallocation
- *find* und logical indexing
- Funktion *strfind*
- Grafik als PDF Datei speichern

Preallocation

Ein Beispiel aus *doc preallocation* zeigt den Geschweindigkeitsunterschied der Berechnungen:

Falsch:

```
tic
x1 = 0;
```

```
for k = 2:1000000
   x1(k) = x1(k-1) + 5;
end
toc
```

Richtig:

```
tic
x2 = zeros(1, 1000000);
for k = 2:1000000
   x2(k) = x2(k-1) + 5;
end
toc
```

Falls möglich:

```
tic
y = (0:1000000-1) * 5;
toc
```

find und logical indexing

```
x = rand(20, 1);

% Vektor mit logicals (true/false)
x < 0.3

% Indizes der Einträge, die die Bedingung erfüllen
find(x < 0.3)

% "Logical Indexing": Werte der Einträge, die die Bedingung erfüllen
x(x < 0.3)

% weitere Anwendungen
sum(x < 0.3)                    % absolute Anzahl der Einträge, die die Bedingung erfüllen
sum(x < 0.3) / length(x) * 100  % relative Anzahl in Prozent

x(x < 0.1) = 0;                 % Werte, die die Bedingung erfüllen, gleich Null setzen
x(x < 0.1) = NaN;               % Werte können als Not-a-Number (NaN) definiert werden,
                                % um von Berechnungen ausgeschlossen zu werden.
```

strfind

Gibt die Indizes des Patern im String zurück.

```
szTest = 'Das hier ist ein Beispiel!';

strfind(szTest, 'e')
strfind(szTest, 'ei')
strfind(szTest, 'ein')
```

print auch als PDF

```
% Plot erstellen
hFig = figure;
plot(rand(1000,1), 'x')

% Plot als eps, png und pdf Datei abspeichern
print(hFig, '-depsc2', 'grafik.eps');     % speichern als EPS
print(hFig, '-dpng',   'grafik.png');     % speichern als PNG
print(hFig, '-dpdf',   'grafik.pdf');     % speichern als PDF
```

Tutorial 5.2

- *csvwrite* / *csvread*
- Tabellen in Matlab
- *tablewrite* / *tableread*

CSV Dateien lassen sich von vielen Programmen einlesen und abspeichern. Sie stellen eine Möglichkeit für den Datenaustausch dar.

```
m = eye(5);
csvwrite('test.csv',m);        % schreiben
m_new = csvread('test.csv');   % lesen
```

Achtung: nur "5 signifikante Ziffern" werden berücksichtig!

```
m1 = rand(5);
csvwrite('test.csv', m1)
m1_new = csvread('test.csv');

m2 = round(rand(5) * 10^7);
csvwrite('test.csv', m2)
m2_new = csvread('test.csv');
```

Gegebenenfalls *dlmwrite* / *dlmread* verwenden.

Tabellen in Matlab

```
LastName      = {'Smith'; 'Johnson'; 'Williams'; 'Jones'; 'Brown'};
Age           = [38; 43; 38; 40; 49];
Height        = [71; 69; 64; 67; 64];
Weight        = [176; 163; 131; 133; 119];
BloodPressure = [124 93; 109 77; 125 83; 117 75; 122 80];

T = table(Age, Height, Weight, BloodPressure, ...
    'RowNames', LastName)
summary(T)
```

```
writetable(T, 'test.csv')
T_new = readtable('test.csv');
```

Fazit:

Für einen Dateiaustausch zwischen Excel/LibreOffice eignen sich csv-Dateien. Sofern Daten nur mit Matlab gespeichert und auch in Matlab wieder eingelesen werden, sollte man mat-Dateien verwenden da dort die Variablen nicht verändert werden.

Tutorial 5.3

- *pathtool* und PATH

Matlab findet Dateien nur wenn sie in dem "current folder" oder im Pfad (bzw. PATH) sind. Hiermit können die Ordner in PATH verändert werden:

```
pathtool
```

Man sollte nicht zu viele Ordner in seinem PATH haben, da es sonst möglich ist, dass man wenn man an einem Projekt arbeitet, auf Funktionen und Skripte aus einem anderen Projekt zugreift, die den selben Namen haben.

Ordner **nicht** im PATH:

```
mkdir test
m = rand(5);
save(fullfile('test','data.mat'))

load('data.mat')     % FEHLER: Ordner "test" ist nicht in PATH
load(fullfile('test','data.mat'))
load(fullfile(pwd, 'test','data.mat'))
```

Ordner PATH hinzufügen:

```
addpath('test')      % Fügt den Ordner "test" dem PATH hinzu
                     % ACHTUNG: dies muss nach jedem Neustart von Matlab wiederholt werden

load('data.mat')     % nun wird die Datei "data.mat" gefunden
```

Tutorial 5.4

- komplexe Zahlen

```
cNum1 = 3 + 4i;
cNum2 = 3 + 4j;
cNum3 = 3 + 4 * 1j;
cNum4 = sqrt(-1);
```

Daher: Niemals i oder j als Laufvariablen in einer Schleife verwenden (besser: n, k, nn, kk etc.).

Funktionen, um mit komplexen Zahlen zu arbeiten.

```
real(cNum1)        % Realteil
imag(cNum1)        % Imaginärteil
abs(cNum1)         % Betrag
```

```
angle(cNum1)        % Winkel
conj(cNum1)         % komplex Konjugieren
```

Vorsicht beim Transponieren von komplexen Zahlen (()' vs. ().'):

```
cMat = (eye(2,1)*1i) + rand(2,1);

cMat'
cMat.'
```

Tutorial 5: Aufgabe

Quellcode vervollständigen, um die untere Grafik zu erzeugen.

```
x = linspace(0, 8*pi, 100);
y = cos(x) + 1i * sin(x);
plot3( ...
```