MatLab Programming Fundamentals

guarantor:	Maroš Tunák
tel.:	3465
e-mail:	*maros.tunak@tul.cz*

Course objectives

The aim of the course is to acquire basics knowledge and skills of students the MatLab program. At the end of the course students will be able to use MatLab for their own work and will be ready to deepen their programming skills in MatLab.

MatLab Programming Fundamentals

time requirements:	0p+2c
credits:	4
exercises:	Monday 10:40-12:15; 12:30-14:05 (B-PC2, Tunák M.) Tuesday 08:50-10:25; 10:40-12:15 (B-PC2, Tunák M.)
consultation:	Wednesday 10:40-12:15 (E-KHT)

Requirements on student/graded credit

1. participation in exercises (max. 3 absences)
2. elaboration of semester work (after approval of the semester work, you can attend a practical demonstration)
3. practical demonstration of acquired skills (there will be 1-2 examples to solve; elaboration time 1 hour; you can use any materials ...)

Content

IS/STAG Syllabus

1. Getting started with Matlab. Working environment, windows, paths, basic commands, variables. Loading, saving and information about variables. Help.

2. Mathematics with vectors and matrices. Creating vectors and matrices. Indexing. Special matrices. Matrix operations. Element by element operations. Relational operations, logical operations, examples and tricks.

3. Control flow. Loops, conditional statements, examples.

4. Script m-files, Function m-files.

5. Visualisation. Two-dimensional graphics. Three-dimensional graphics.

6. Graphical user interface.

7.-10. Statistics and Machine Learning Toolbox. Basics of statistical data processing, exploratory data analysis, descriptive statistics, data visualisation, hypothesis testing, confidence intervals, regression analysis, control charts.

11.-13. Solution of practical problems in textile and industrial engineering.

Literature

Recommended

MathWorks. *Getting Started with MATLAB.* [Online]. Dostupné z: https://www.mathworks.com/help/matlab/getting-started-with-matlab.html

Study materials

http://elearning.tul.cz

Installation

http://liane.tul.cz/cz/software/MATLAB

m-files. Script m-files, Function m-files.

m-files

MatLab provides a powerful programming language and an interactive computing environment. So far we have been working with MatLab in interactive mode, i.e. all commands were written in the command window and immediately executed after entering. However, we often need to repeat certain command sequences. For this purpose, we can write a set of commands into a file, which we then run as any MatLab command or function.

Programs or sequences of commands and functions can be stored in so-called m-files (*m-files*). These are text files that are saved with the `*.m` extension. You can use any text editor to create a text file; MatLab, invoked by the `edit` command.

m-files

There are two types of program files (m-files):

- m-files, scripts that do not accept input arguments and do not return output arguments. They only work with data in the workspace.
- m - files, functions that can accept input arguments and return output arguments. Internal variables are local to the function (they are not created in the workspace).

Script m-files

m-files, scripts - command sequences stored in files

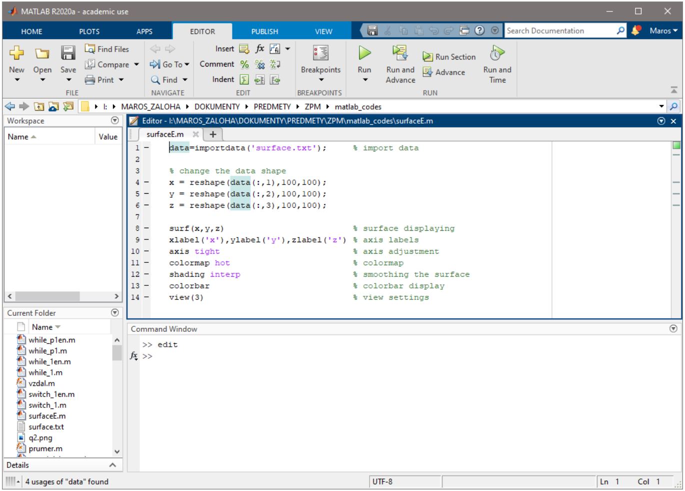
- files with extension `*.m`
- plain text (ASCII) files
- they may refer to other m-files
- scripts are executed by typing the file name in the command window, or by using the *Run* icon in the editor (▷ or the F5 key)
- once executed, the commands written on each line are executed sequentially
- all variables we create in the script are stored in the workspace

Script m-files

- Create an example script from the previous exercise: display surface of textile fabric (10 × 10 mm with division step 0.1 mm) captured by Talysurf contactless laser profilometer, where the profile is stored in the form of X, Y, Z coordinates in the text file *surface.txt*.
- open the MatLab editor (Editor could be docked (Fig. 1) or undocked (Fig. 2))

```
>> edit
```

Script m-files



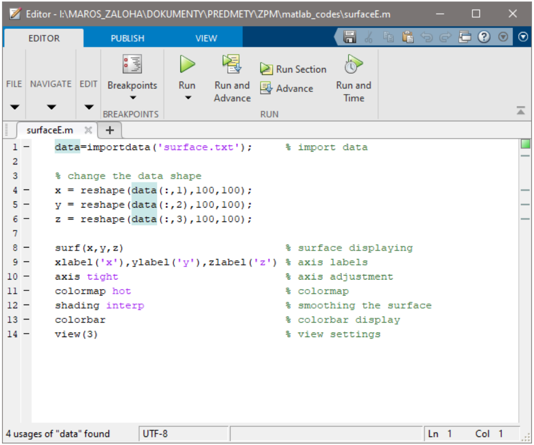
```
data=importdata('surface.txt');      % import data

% change the data shape
x = reshape(data(:,1),100,100);
y = reshape(data(:,2),100,100);
z = reshape(data(:,3),100,100);

surf(x,y,z)                          % surface displaying
xlabel('x'),ylabel('y'),zlabel('z')  % axis labels
axis tight                           % axis adjustment
colormap hot                         % colormap
shading interp                       % smoothing the surface
colorbar                             % colorbar display
view(3)                              % view settings
```

```
>> edit
>>
```

4 usages of "data" found

Figure: Editor docked

Script m-files



```
data=importdata('surface.txt');      % import data

% change the data shape
x = reshape(data(:,1),100,100);
y = reshape(data(:,2),100,100);
z = reshape(data(:,3),100,100);

surf(x,y,z)                          % surface displaying
xlabel('x'),ylabel('y'),zlabel('z')  % axis labels
axis tight                           % axis adjustment
colormap hot                         % colormap
shading interp                       % smoothing the surface
colorbar                             % colorbar display
view(3)                              % view settings
```

Figure: Editor undocked

Script m-files

- save the script under the name *surfaceE.m* - (EDITOR - Save)
- and we write a sequence of commands

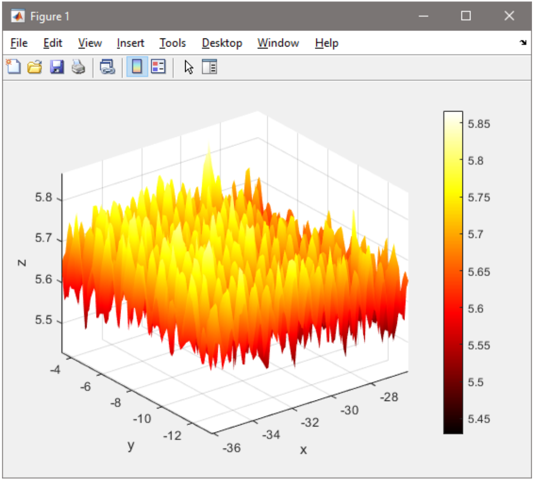
Script m-files

```
data=importdata('surface.txt');      % import data

% change the data shape
x = reshape(data(:,1),100,100);
y = reshape(data(:,2),100,100);
z = reshape(data(:,3),100,100);

surf(x,y,z)                          % surface displaying
xlabel('x'),ylabel('y'),zlabel('z')  % axis labels
axis tight                           % axis adjustment
colormap hot                         % colormap
shading interp                       % smoothing the surface
colorbar                             % colorbar display
view(3)                              % view settings
```

- run the file by the *Run* icon, or by pressing the F5, or by typing the file name in the command window (files *surfaceE.m*, *surface.txt* must be in the same directory and set as the current directory)

```
>> surfaceE
```

Script m-files



Script m-files

- Create an example script from the previous exercise: Draw the surface of the sphere with the center $[x_0 = 0, y_0 = 0, z_0 = 0]$ and the radius $r = 1$. Parametric expression of spherical surface is given by:

$$
\begin{aligned}
x &= x_0 + r\ cos\phi\ sin\theta \\
y &= y_0 + r\ sin\phi\ sin\theta \\
z &= z_0 + r\ cos\theta
\end{aligned}
$$

  for $0 < \phi \leqslant 2\pi,\ 0 \leqslant \theta \leqslant \pi$. In the same figure draw another sphere with the radius $r = 2$ centred at

$$[x_0 = 1, y_0 = 1, z_0 = 1]$$

- open the MatLab editor

```
>> edit
```

- save the script under the name *ball.m* - (EDITOR - Save)
- and we write a sequence of commands

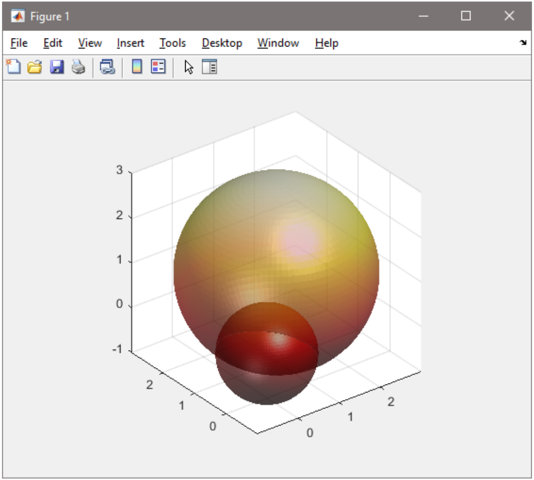
Script m-files

```
N = 50;                             % number of elements
theta = linspace(0,pi,N);           % theta angle vector
phi = linspace(0,2*pi,2*N);         % phi angle vector
[th, ph] = meshgrid(theta,phi);     % meshgrid of coordinates
r=1;                                % radius
x0=0;y0=0;z0=0;                     % center
                                    % parametric expression
x=x0+r*sin(th).*cos(ph);
y=y0+r*sin(th).*sin(ph);
z=z0+r*cos(th);

surf(x,y,z);                        % displaying of surface
hold on

% second ball
r=2;                                % radius
x0=1;y0=1;z0=1;                     % center
x=x0+r*sin(th).*cos(ph);            % parametric expression
y=y0+r*sin(th).*sin(ph);
z=z0+r*cos(th);
surf(x,y,z);                        % displaying of surface

axis equal,shading interp,camlight  % display settings
alpha(0.5)                          % transparency
```

- run the file by the *Run* icon, or by pressing the F5 key or by typing the file name in the command window (file *ball.m* must be in the current directory)

```
>> ball
```

Script m-files



Function m-files

m-files, functions - command sequences stored in files

- functions are files that can accept input arguments
- defined variables are local
- functions are files that can return output arguments
- unlike the script, the first line of function has a fixed syntax that looks generally

```
function [out1,..,outtM]=fc_name(in1,..,intN)
% help comment line
```

Function m-files

- the function is then saved to a file with the same name as the first line of the function `fc_name.m`.
- the input parameters are passed to the function `in1,..,inN`
- the function has its own workspace, all variables that are created in the function body are local variables and are not stored in the workspace
- the output of the function are output variables `out1,..,outM`
- it is useful to insert several information comment lines before the function body itself, which are then invoked by the command `help fc_name`.

Function m-files

- example of creating the `average.m` function to calculate the arithmetic mean of data:

```
function m=average(x)
% average - calculate the arithmetic mean of the data
% x - is a row or column data vector
n=length(x);
s=sum(x);
m=s/n;
```

- we want to calculate the arithmetic mean of the data

```
>> data=[15.3 17.1 17.3 12.5 20]
data =
   15.3000    17.1000    17.3000    12.5000    20.0000
```

- and call the function

```
>> m=average(data)
m =
   16.4400
```

Function m-files

- the current directory must contain the created function, otherwise MatLab will not find the function, or
- the directory containing the created function will be included in directories that MatLab automatically searches (HOME - Set Path)
- we try the commands `lookfor`

```
>> lookfor average
average - calculate the arithmetic mean of the data
```

- and `help`

```
>> help average
  average - calculate the arithmetic mean of the data
  x - is a row or column data vector
```

Examples for practice

Examples for practice

1. Write a script to draw a force-deformation curve of the electric fence that is on the first sheet (*sample1*) of the file *fences.xlsx*
    - use and study the command `xlsread` to retrieve data from the file
    - insert axis labels
2. Create a function (*distan.m*) to calculate the distance of two points in the plane $[x, y]$

Solution