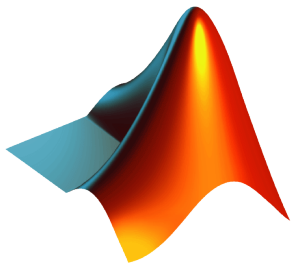
MatLab Programming Fundamentals

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

Control flow. Loops, conditional statements, examples.

if

if - decision making structures. Conditional statements enable you to select at run time which block of code to execute. The simplest conditional statement is an `if`.

Syntax

```
» if      expression 1
          statements 1
elseif    expression 2
          statements 2
else      statements 3
» end
```

if

- **Example:** in MatLab editor we create a script named `if_1en.m` and write it

```
1  clear,clc
2  a=randi(100,1) % returns a pseudorandom scalar integer ...
       between 1 and 100
3  if a<20          % if the condition is true
4      disp('number a is less than 20') % do the command
5  end
```

save and run.

```
a =
    14
number a is less than 20
```

The script generates a pseudo-random integer number a uniformly distributed in the interval 1 to 100. If the number a is less than 20, the text *"number a is less than 20"* is displayed (conditional statement `if`). Otherwise nothing will be done.

if

- **Example:** we will add the option if the condition is false (`else`). We create a script named `if_2en.m` and write it

```
1  clear,clc
2  a=randi(100,1) % returns a pseudorandom scalar integer ...
       between 1 and 100
3  if a<20          % if the condition 1 is true
4      disp('number a is less than 20') % do the command
5  else             % else (if the condition 1 is false)
6      disp('number a is greater than 20') % do the command
7  end
```

save and run.

```
a =
    44
number a is greater than 20
```

The script generates a pseudo-random integer number `a` uniformly distributed in the interval 1 to 100. If the number `a` is less than 20, the text *"number a is less than 20"* is displayed (conditional statement `if`), and if the condition is not true, the text *"number is greater than 20"* is displayed (`else`).

if

- `if` statement can be followed by one (or more) optional `elseif`, which is very useful to test various conditions. We create a script named `if_3en.m` and write it

```
1  clear,clc
2  a=randi(100,1)       % returns a pseudorandom scalar ...
       integer between 1 and 100
3  if a<33              % if the condition 1 is true
4      disp('small')    % do the command
5  elseif a<66          % if the condition 2 is true
6      disp('medium')   % do the command
7  else                 % else
8      disp('large')    % do the command
9  end
```

save and run.

```
a =
    65
medium
```

The script generates a pseudo-random integer number `a` uniformly distributed in the interval 1 to 100. If the number `a` is smaller thane 33, the text *"small"* (`if`) is displayed. In case, that number `a` is smaller than 66, the text *"medium"* is displayed(`elseif`). Otherwise the text *"large"* is displayed (`else`).

switch, case

- Alternatively, when you want to test for equality against a set of known values, use a `switch` statement together with `case` statement. For example, a script `switch_1en.m`

```
1  clear,clc
2  gradeECTS='H'
3  switch(gradeECTS)
4      case 'A'
5          disp('excellent');
6      case 'B'
7          disp('very good');
8      case 'C'
9          disp('good');
10     case 'D'
11         disp('satisfacory');
12     case 'E'
13         disp('sufficient');
14     case 'F'
15         disp('fial');
16     otherwise
17         disp('incorrectly enterd grade');
18 end
```

switch, case

save and run.

```
gradeECTS =
    'B'
very good
```

or

```
gradeECTS =
    'H'
incorrectly enterd grade
```

In general, when you have many possible discrete, known values, `switch` statement together with `case` statements are easier to read than `if`. However, you cannot test for inequality between `switch` and `case` values.

break

The `break` statement terminates execution of `for` or `while` loop. Statements in the loop that appear after the `break` statement are not executed.

- **Example:** in MatLab editor we create a script named `break_1en.m`, where we create the variable `x=1` and in the body of the loop the variable `x` will increase by one. The loop will repeat until `x<12`. The command `break` terminates the loop if the condition `x>7` is true.

```
1  clear,clc,close all
2
3  x=1;
4  while x<12
5      disp(x)
6      x=x+1;
7      if x>7
8          break
9      end
10 end
```

break

save and run.

```
1
2
3
4
5
6
7
```

continue

The `continue` statement is used for passing control to next iteration of `for` or `while` loop. It works somewhat like `break` statement. Instead of forcing termination, however, "continue" forces the next iteration of the loop to take place, skipping any code in between.

- **Example:** in MatLab editor we create a script named `continue_1en.m`, where we create the variable `x=1` and in the body of the loop the variable `x` will increase by one. The loop will repeat until `x<12`. Command `continue` skips the iteration if the condition `x==7` is true.

```
1  clear,clc,close all
2
3  x=1;
4  while x<12
5      if x==7
6          x=x+1;
7          continue
8      end
9      disp(x)
10     x=x+1;
11 end
```

continue

save and run.

```
1
2
3
4
5
6
8
9
10
11
```

Examples for practice

Examples for practice

1. Prepare a script (`if_1pen.m`) that contains any two numbers A and B. Use `if` to test if A is greater than, less than or equal to B, and display it in the command window.
2. Plot function $f(x)$, for which:

$$f(x) = \begin{cases} -2x & x \leqslant 0 \\ x & 0 < x \leqslant 1 \\ x^2 & x > 1 \end{cases}$$

x is in the range $< -2, 3 >$. Use the `for` loop and the `if` commands. Create script (`if_2pen.m`).

Solution