# 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.

# Control flow

When solving technical tasks, you may need to execute the same command or set of commands or code block several times in a row. Generally, commands are executed sequentially, the first command being executed first, then the second, and so on. A loop (cycle) allows us to execute commands or a group of commands multiple times. MatLab provides the following types of loops (cycles) for processing repeat requests:

- `for`
- `while`

The main use of loops is when writing m-files, but can also be used in interactive mode in the command window.

# for

for - loop for a predetermined (fixed) number of repetitions, iterations, command, or command group

Syntax

```
» for   variable = values
        statement 1
        statement 2
        ⋮
        statement n
» end
```

number of repetitions is given by *values*, which can be in the form of a vector, most often in the form:

| | |
|---|---|
| first:step:last | vector with linear series (vector of elements with value *first*, incrementing by *step*, until it reaches *last* |
| first:last | *step* = 1 |
| first:-step:last | negative step |
| [values] | specific values |

# for

- **Example:** type the following in the command window:

```
>> for a=45:62
disp(a)
end
```

```
45

46

47

48

...
```

The control variable a will gradually take values from 45 to 62 in steps of 1 in the loop. The value of the control variable (function disp) should be displayed in the loop body. In the command window, separate commands are written on separate lines, and the loop does not begin until the end keyword is specified.

# for

- **Example:** type the following in the command window:

```
>> for b=1:-0.1:0
disp(b)
pause(1)
end
```

```
1

0.9000

0.8000

0.7000

...
```

The control variable `b` will take values from 1 to 0 in steps of -0.1. The value of the control variable (the first statement) should be displayed in the cycle body and wait for one second (the second statement, the `pause` function). In the command window, separate commands are written on separate lines, and the cycle does not begin until the `end` keyword is specified.

# for

- **Example:** type the following in the command window:

```
>> for c=[15 23.1 9 17.2 44.5 12]
[c c^2]
pause(1)
end
```

```
ans =
    15   225

ans =
   23.1000   533.6100

ans =
     9    81

...
```

the control variable `c` will get values 15, 23.1, 9, ... in the loop. A matrix (the first statement) is created in the loop, which is composed of two values `c` and `c^2` and waits for one second (the second statement). In the command window, separate commands are written on separate lines, and the cycle does not begin until the `end` keyword is specified.

# for

- **Example:** in the MatLab editor we create a script called `for_1en.m`, in which we gradually fill a vector of 15 elements with $i$-th square root of number 100, where $i$ is the sequence number of the vector element. In mathematical notation $x(i) = \sqrt[i]{(100)}$ and in the form of code:

```
clear,clc
for i=1:15
    x(i)=100^(1/i);
end
x
```

save the code and execute with the Run icon (F5) or by typing the script name (without extension) in the command window. The loop is executed and the result is displayed.

```
>> for_1eng
x =
  100.0000   10.0000    4.6416    3.1623    2.5119    2.1544   ...
     1.9307    1.7783    1.6681    1.5849    1.5199    1.4678   ...
      1.4251    1.3895    1.3594
```

`clear` and `clc` in the first line of the script clears all variables and clears the command window.

# for

- **Example:** in the MatLab editor, we create a script named `for_2eng.m` to create a table with values from 1 to 10 in the first column and cumulative sums in the second column.

```
clear,clc
result=[];
for k=1:10
    s=sum(1:k);
    result=[result; k s];
end
result
```

save the code and execute. The loop is executed and the result is displayed.

```
>> for_2eng
result =
     1     1
     2     3
     3     6
     4    10
     5    15
     ...
```

# for

- **Example:** several nested loops can be used, they are indented for clarity, the loop create a matrix of size $3 \times 5$ with numbers from 1 to 15 is given (`for_3eng.m`):

```
clear,clc
counter=1;
for i=1:3
    for j=1:5
        M(i,j)=counter
        counter=counter+1;
    end
end
M
```

save the code and execute. The loop is executed and the result is displayed.

```
M =
     1     2     3     4     5
     6     7     8     9    10
    11    12    13    14    15
```

# for

- **Example:** write a script (`force_def_all.m`) to draw and compare force-deformation curves of electric fences on the nine sheets (*sample1*, *sample2*,..., *sample9*) of the *fences.xlsx* file.

```
clear,clc,close all

figure,hold on
for i=1:9
    data = xlsread('fences.xlsx',['sample' num2str(i)],'A:B');
    plot(data(:,2),data(:,1),'.-')
    disp(['sample ' num2str(i) ' is done'])
    legendDescription{i} = ['sample ' num2str(i)];
end
xlabel('Deformation [mm]')
ylabel('Force [N]')
grid on
legend(legendDescription)
```

save and run.

# for

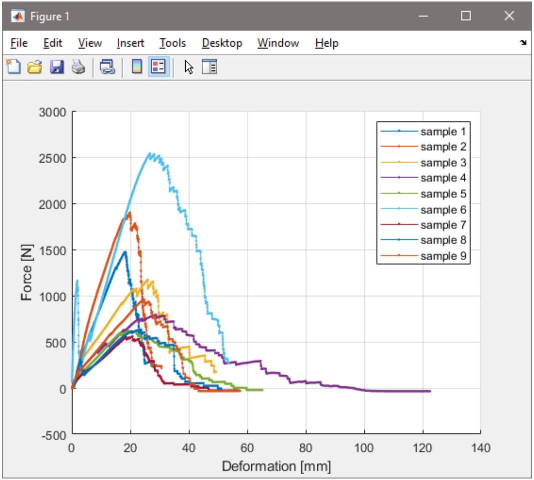
Figure 1
File Edit View Insert Tools Desktop Window Help
Force [N]
Deformation [mm]
3000
2500
2000
1500
1000
500
0
-500
0
20
40
60
80
100
120
140
sample 1
sample 2
sample 3
sample 4
sample 5
sample 6
sample 7
sample 8
sample 9

## while

while - loop for repeated execution of a command, iteration, or group of commands while the condition is true

Syntax

```
» while    expression
           statement 1
           statement 2
           ⋮
           statement n
» end
```

The `while` loop repeatedly executes program statement(s) as long as the expression remains true. An expression is true when the result is nonempty and contains all nonzero elements (logical or real numeric). Otherwise, the expression is false.

# while

- **Example:** in the MatLab editor we create a script named `while_1eng.m`, where we create the variable $x=1$ and the variable `x` will will increase by one in the loop body. The loop is repeated until $x<12$.

```
clear,clc,close all

x=1;
while x<12
    disp(x)
    x=x+1;
end
```

save and run. The loop is executed and the result is displayed.

```
>> while_1eng
     1
     2
     3
    ...
    10
    11
```

# Examples for practice

# Examples for practice

1. Measurements yielded the frequencies $f_i = [0, 2, 5, 7, 6, 9, 13, 8, 5, 3, 0]$ in $i$th class $i = 1, 2, ..., 11$. Use `for` to create a table with columns classes $i$, frequencies $f_i$, cumulative frequencies $F_i$, relative frequencies $f_i/n$ and relative cumulative frequencies $F_i/n$, and create m-script(`for_p1eng.m`).
2. Use the `for` loop to calculate the factorial $i = 1, 2, ..., 10$ and create a table with $i$ in the first column and a factorial of $i!$ in the second column and create m-script(`for_p2eng.m`).
3. Use the `while` loop to create a table with $x = 1, 2, 3...$ in the first column ($x$ will increase by 1 in the loop body), in the second column $x^2$ and third $x^3$. Stop the loop until $x^3 < 2000$ is reached and create m-script(`while_p1eng.m`).

## Solution

# Examples for practice

1. for_p1eng.m

```
tab =
    1.0000         0         0         0         0
    2.0000    2.0000    2.0000    0.0345    0.0345
    3.0000    5.0000    7.0000    0.0862    0.1207
    4.0000    7.0000   14.0000    0.1207    0.2414
    5.0000    6.0000   20.0000    0.1034    0.3448
    6.0000    9.0000   29.0000    0.1552    0.5000
    7.0000   13.0000   42.0000    0.2241    0.7241
    8.0000    8.0000   50.0000    0.1379    0.8621
    9.0000    5.0000   55.0000    0.0862    0.9483
   10.0000    3.0000   58.0000    0.0517    1.0000
   11.0000         0   58.0000         0    1.0000
```

## Examples for practice

2. for_p2eng.m

```
tab =
           1           1
           2           2
           3           6
           4          24
           5         120
           6         720
           7        5040
           8       40320
           9      362880
          10     3628800
```

## Examples for practice

3. `while_p1eng.m`

```
tab =
          1          1          1
          2          4          8
          3          9         27
          4         16         64
          5         25        125
          6         36        216
          7         49        343
          8         64        512
          9         81        729
         10        100       1000
         11        121       1331
         12        144       1728
```