A0B17MTB – Matlab

Part #3

Miloslav Čapek
miloslav.capek@fel.cvut.cz

Filip Kozák, Viktor Adler, Pavel Valtr

Department of Electromagnetic Field
B2-626, Prague

Learning how to …

```
ResTable.data1(...
   PsoData.cond{crt}(spr,2),...
   PsoData.cond{crt}(spr,3) ...
   ) = ...
 bestPersDim(bestGlobNum, crt);
```

Indexing

Size and type of data

Output format

`Matlab Editor`

Department of Electromagnetic Field, CTU FEE, miloslav.capek@fel.cvut.cz

Indexing in Matlab

- now we know all the stuff necessary to deal with indexing in Matlab
- mastering indexing is crucial for efficient work with Matlab!!!
- up to now we have been working with entire matrices, quite often we need, however, to access individual emenets of matrices
- two ways of accessing matrices / vectors are distinguished
  - access using round brackets „`( )`“
    - refers to position of elements in a matrix
  - access using square brackets „`[ ]`“
    - refers to content of a matrix

Department of Electromagnetic Field, CTU FEE, miloslav.capek@fel.cvut.cz

Indexing in Matlab

600 s

- let's consider following triplet of matrices
  - execute individual commands and find out their meaning
  - start from inner part of the commands
  - note the meaning of the keyword `end`

$$\mathbf{N}_1 = \begin{pmatrix} -5 \\ 0 \\ 5 \end{pmatrix} \qquad \mathbf{N}_2 = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 4 & 6 & 8 & 10 \\ 2 & 3 & 5 & 7 & 11 \end{pmatrix} \qquad \mathbf{N}_3 = \begin{pmatrix} 11 & 12 & 13 & 14 \\ 22 & 24 & 26 & 28 \\ 33 & 36 & 39 & 42 \\ 44 & 48 & 52 & 56 \end{pmatrix}$$

```
>> N1 = (-5:5:5)';N2 = [1:5;2:2:10;primes(11)];N3 = (1:4)'*(11:14);
```

```
>> N1
>> N1(1:3)
>> N1([1 2 3])
>> N1(1:2)
>> N1([1 3])
>> N1([1 3]')
>> N1([1 3])'
>> N1([1; 3])
```

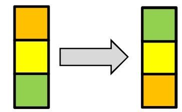
```
>> N2(1, 3)
>> N2(3, 1)
>> N2(1, end)
>> N2(end, end)
>> N2(1, :)
>> N2(1, :)'
>> N2(:, 2)
>> N2(:, 3:end)
```

```
>> N3(2:3, [1 1 1]) % like repmat
>> N3(2:3, ones(1,3))
>> N3(2:3, ones(3,1))
>> N3([N2(2,1:2)/2 4], [2 3])
>> N3([1 end], [1:4 1:2:end])
>> N3(:, :, 2) = magic(4)
>> N3([1 3], 3:4, 3) = ...
       [1/2 -1/2; pi*ones(1, 2)]
```

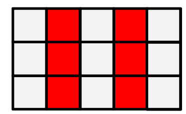
Indexing in Matlab

420 s

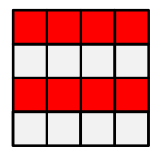
- remember the meaning of `end` and the usage of colon operator “`:`“
- try to:
  - flip the elements of the vector **N1**



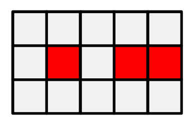
    - without using `fliplr` / `flipud` functions
  - select only the even columns of **N2**



  - select only the odd rows of **N3**



  - 2nd, 4th and 5th column of **N2**’s 2nd row



  - create matrix **A** (4x3) containing numbers 1 to 12 (row-wise, from left to right)

Department of Electromagnetic Field, CTU FEE, `miloslav.capek@fel.cvut.cz`

Indexing in Matlab

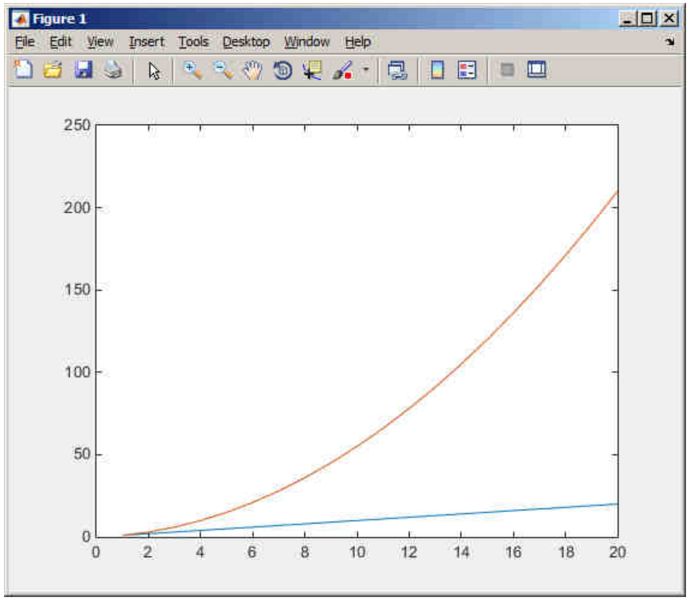
300 s

- calculate cumulative sum **S** of a vector **x** consisting of integers from 1 to 20
  - search Matlab help to find appropriate function (*cumulative sum*)

$$\mathbf{x} = \begin{pmatrix} 1 & 2 & \ldots & 20 \end{pmatrix}$$

$$S = \begin{pmatrix} 1 & 1+2 & \ldots & 1+2 \cdots + 20 \end{pmatrix}$$

  - calculate cumulative sum **L** of even elements of the vector **x**
  - what is the value of the last element of the vector **L**?



Indexing in Matlab

150 s

- which one of the following returns corner elements of a matrix A (10x10)?

```
>> A([1,1], [end,end])                           % A.
>> A({[1,1], [1,end], [end,1], [end,end]}) % B.
>> A([1,end], [1,end])                           % C.
>> A(1:end, 1:end)                               % D.
```

Deleting elements of a matrix

- empty matrix is a crucial point for deleting matrix elements

```
>> T = []
```

- we want to:
  - remove 2nd row of matrix **A**

```
>> A(2, :) = []
```

  - remove 3rd column of matrix **A**

```
>> A(:, 3) = []
```

  - remove 1st, 2nd a 5th column of matrix **A**

```
>> A(:, [1 2 5]) = []
```


Department of Electromagnetic Field, CTU FEE, miloslav.capek@fel.cvut.cz

Adding and replacing elements of a matrix

- we want to replace:
  - 3rd column of matrix **A** (of size M×N) by a vector **x** (length N)

```
>> A(:, 3) = x
```

  - 2nd, 4th a 5th row of matrix **A** by three rows of matrice **B** (number of columns of both **A** and **B** is the same)

```
>> A([2 4 5], :) = B(1:3, :)
```

- we want to swap
  - 2nd row of matrix **A** and 5th column of matrix **B** (number of columns of **A** is the same as number of rows of **B**)

```
>> A(2, :) = B(:, 5)
```

- remember that always the size of matrices have to match!

Deleting, adding and replacing matrices

420 s

- which of the following deletes the first and the last column of matrix **A** (6×6)?
  - create your own matrix and give it a try

```
>> A[1, end]      = 0  % A.
>> A(:, 1, end)   = [] % B.
>> A(:, [1:end]) = [] % C.
>> A(:, [1 end]) = [] % D.
```

- replace the 2nd, 3rd and 5th row of matrix **A** by the first row of matrix **B**
  - assume the number of columns of matrices **A** and **B** is the same
  - consider the case where **B** has more columns than **A**
  - what happens if **B** has less columns than **A**?

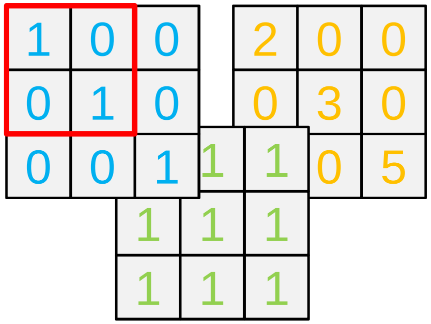
Department of Electromagnetic Field, CTU FEE, miloslav.capek@fel.cvut.cz

Matrix creation, element replacement

300 s

- create following 3D array

$$\mathbf{M}(:,:,1)=\begin{pmatrix}1&0&0\\0&1&0\\0&0&1\end{pmatrix},\quad \mathbf{M}(:,:,2)=\begin{pmatrix}1&1&1\\1&1&1\\1&1&1\end{pmatrix},\quad \mathbf{M}(:,:,3)=\begin{pmatrix}2&0&0\\0&3&0\\0&0&5\end{pmatrix}$$

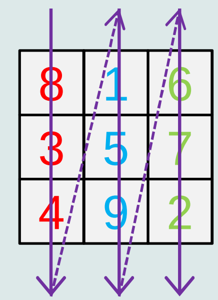


- replace elements in the first two rows and columns of the first sheet of the array (i.e. the matrix [1 0; 0 1]) with `NaN` elements

Linear indexing

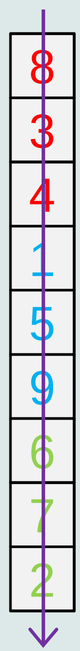
- elements of an array of arbitrary number of dimensions and arbitrary size can be referred to using single index
  - indexing takes place along the main dimension (column-wise) than along the secondary dimension (row-wise) etc.

```
>> A = magic(3)

A =

     8     1     6
     3     5     7
     4     9     2

>> A(:)
```




```
>> A = magic(3)
```

```
>> A(:)
```

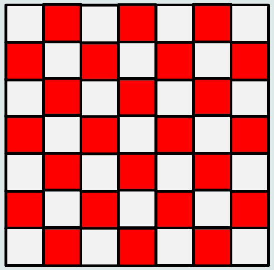



```
ans =

     8
     3
     4
     1
     5
     9
     6
     7
     2
```

Linear indexing - application

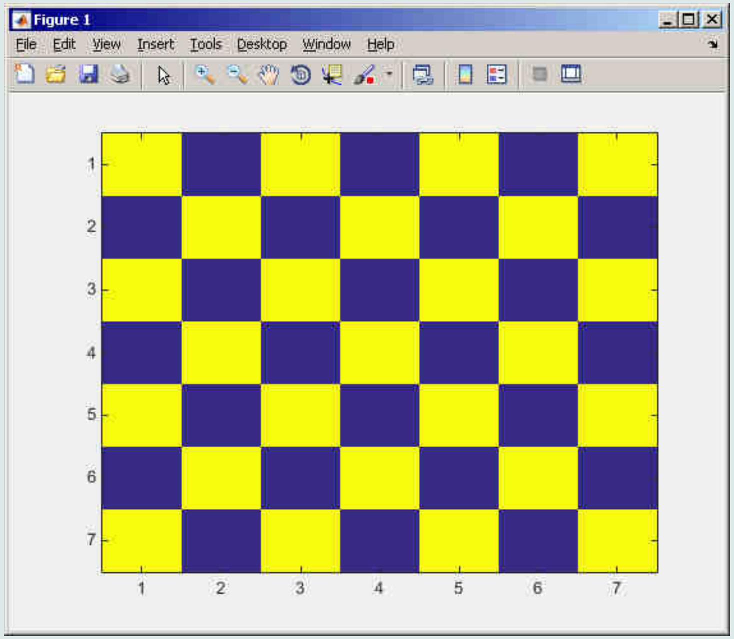
- let's consider following matrix:

```
>> MAT = ones(7);
```



  - we set all the red-highlighted elements to zero:

```
>> MAT(2:2:end) = 0
>> imagesc(MAT);
```



Linear indexing – `ind2sub`, `sub2ind`

- `ind2sub`: recalculates linear index to subscript corresponding to size and dimension of the matrix
  - applicable to an array of arbitrary size and dimension

1	4	7
2	5	8
3	6	9

1,1	1,2	1,3
2,1	2,2	2,3
3,1	3,2	3,3

```
>> ind = 3:6;
>> [rw, col] = ind2sub([3, 3], ind)
% rw  = [3 1 2 3]
% col = [1 2 2 2]
```

- `sub2ind`: recalculates subscripts to linear index
  - applicable to an array of arbitrary size and dimension

1,1	1,2	1,3
2,1	2,2	2,3
3,1	3,2	3,3

1	4	7
2	5	8
3	6	9

```
>> ind2 = sub2ind([3, 3], rw, col)
% ind2 = [3 4 5 6]
```

Linear indexing

300 s

- for a two-dimensional array, find a formula to calculate linear index from position given by `row` (row) and `col` (column)
  - check with a matrix A of size 4×4, where
    - `row = [2, 4, 1, 2]`
    - `col = [1, 2, 2, 4]`
  - and therefore
    - `ind = [2, 8, 5, 14]`

```
>> A = zeros(4);
>> A(:) = (1:16)
```


Department of Electromagnetic Field, CTU FEE, `miloslav.capek@fel.cvut.cz`

Function who, whos

- function `who` lists all variables in Matlab Workspace
  - wide variety of options

- function `whos` lists the variable names + dimension, size and data type of the variables or displays content of a file
  - wide variety of options

```
>> whos('-file', 'matlab.mat');
```

```
>> a = 15; b = true;
>> c = 'test'; d = 1 + 5j;
>> who
>> whos
>> Ws = whos;
```


Department of Electromagnetic Field, CTU FEE, miloslav.capek@fel.cvut.cz

Function `what`, `which`, `delete`

- function `what` lists names of all Matlab files in the current folder

```
>> Wt = what;
```

- funkce `which` is able to localize (in this order)
  - `.m` / `.p` / Simulink function
  - Method of Java class
  - Workspace variable
  - arbitrary file, if present in the current folder

```
>> which sin
built-in (C:\Program Files\MATLAB\R2013a\toolbox\matlab\elfun\@double\sin)  % double method
```

- function `delete` deletes
  - files
  - handle objects (e.g. graphical objects)

Department of Electromagnetic Field, CTU FEE, miloslav.capek@fel.cvut.cz

Functions `cd`, `pwd`, `dir`

- function `cd` changes current folder
  - lists current folder when called without a parameter
  - „`cd ..`“ jumps up one directory, „`cd /`“ jumps up to root
- function `pwd` identifies current folder
- function `dir` lists current folder content
- for other functions (`mkdir`, `rmdir`, …) see Matlab Help

Department of Electromagnetic Field, CTU FEE, `miloslav.capek@fel.cvut.cz`

Function `prefdir`

- folder containing preferences, history, and layout files

```
>> folder = prefdir
>> cd(folder);
```

- it is recommended to do not edit any file!

Department of Electromagnetic Field, CTU FEE, miloslav.capek@fel.cvut.cz

Function `memory`, `ver`

- function `memory` displays information on how much memory is available and how much the MATLAB software is currently using

```
>> memory
>> M = memory
```

```
>> memory
Maximum possible array:        4408 MB (4.622e+09 bytes) *
Memory available for all arrays:       4408 MB (4.622e+09 bytes) *
Memory used by MATLAB:         696 MB (7.294e+08 bytes)
Physical Memory (RAM):        3534 MB (3.705e+09 bytes)

*  Limited by System Memory (physical + swap file) available.
```

- function `ver` displays license information
  - Matlab version
  - License number
  - List of toolboxes and their version

```
>> ver
>> V = ver
```

- if you need to know the vesion of Matlab only, use `version`

```
>> V = version
```

Format of command line output

```
>> pi

ans =

    3.1416

>> sin(1.1)

ans =

    0.8912
```

- up to now we have been using basic setup
- Matlab offers number of other options
  - use `format setting`
  - output format does not change neither the computation accuracy nor the accuracy of stored result (`eps`, `realmax`, `realmin`, … still apply)

`setting`	format description
`short`	fixed 4 decimal points are displayed
`long`	15 decimal points for double accuracy, 7 decimal points for single accuracy
`shortE`	floating-point format (scientific notation)
`longE`	-//-
`bank`	Two decimal points only (euro – cents)
`rat`	Matlab attempts to display the result as a fraction
and others	note.: omitting `setting` parameter restors default setup

Format of command line output

240 s

- try following output format settings
  - each format is suitable for different type of problem

```
>> s = [5 1/2 1/3 10*pi sqrt(2)];
>> format long; s
>> format rat; s
>> format bank; s
>> format hex; s
>> format +; s
>> format; s
```

- there exist other formats with slight differences
  - check `doc format`

- later, we will learn how to use formatted conversion into strings (commands `sprintf` a `fprintf`)

Department of Electromagnetic Field, CTU FEE, miloslav.capek@fel.cvut.cz

List of ASCII characters

- ASCII characters used in Matlab
  - All characters to be found on EN keyboard

[	ALT + 91	matrix definition, indexing
]	ALT + 93	-//-
{	ALT + 123	cell elements indexing
}	ALT + 125	-//-
@	ALT + 64	handle (symbolic math)
>	ALT + 62	relation operator
<	ALT + 60	-//-
\	ALT + 92	Matrix division
|	ALT + 124	logical operator
~	ALT + 126	-//-
^	ALT + 94	power

  - for more see: `http://www.asciitable.com/`

Launching external programs

- rarely used

- external programs are launched using the exclamation mark "!"
  - the whole line after the "!" is processed as operation system command

```
>> !calc
```

  - if you don't want to interrupt execution of Matlab by the launch, add "&"

```
>> !calc &
>> !notepad notes.txt &
```

  - it is possible to run Matlab with several ways

```
>> doc matlab Windows
>> doc matlab UNIX
```

Work with files using the prompt

- try the following
  - copy & paste line by line, observe what happens
  - be careful when editing the commands!!!

```
>> mkdir('My_experiment');
>> cd('My_experiment');
>> this_directory = pwd;
>> our_file = 'pathdef.m';
>> our_data = fullfile(matlabroot, 'toolbox', 'local', our_file);
>> copyfile(our_data, this_directory);
>> new_file = 'my_demo.txt';
>> movefile(our_file, new_file);
>> !write my_demo.txt
```


Department of Electromagnetic Field, CTU FEE, miloslav.capek@fel.cvut.cz

Exercise #1

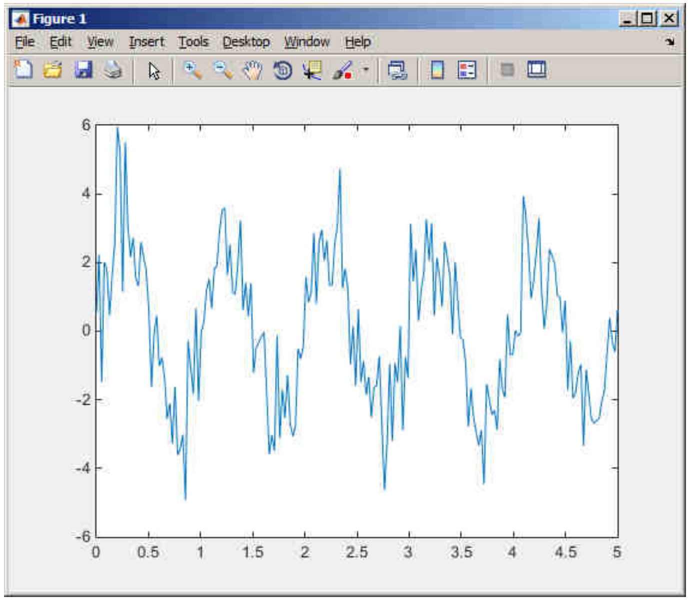
600 s

- consider signal: $s(t)=\sqrt{2\pi}\sin(2\omega_0 t)+n(\mu,\sigma),\quad \omega_0=\pi,$
  where the mean and standard deviation of normal distribution n is:

  `mu` $\mu=0,\quad \sigma=1$ `sigma`

  - create time dependence of the signal spanning N = 5 periods of the signal using V = 40 samples per period
  - one period: $T=1:\quad t\in[k,\mathrm{k}+1],\ k\in\mathbb{Z}^0$ (choose k equal for instance to 0)
  - the function $n(\mu,\sigma)$ has Matlab syntax:

```
>> n = mu + sigma*randn(1, N*V)
```



Department of Electromagnetic Field, CTU FEE, miloslav.capek@fel.cvut.cz

Exercise #2

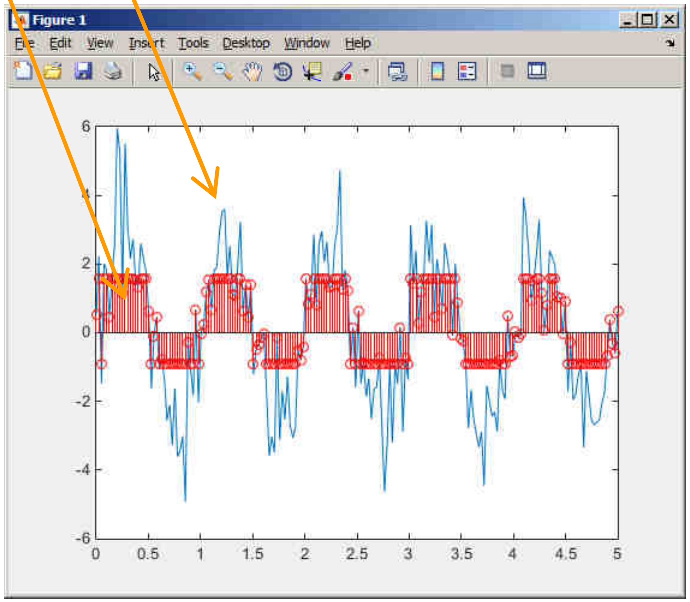
600 s ↑

- apply threshold function to generated signal from the previous exercise to limit its maximum and minimum value:

$$s_{\mathrm{p}}\left(t\right)=\begin{cases} s_{\min} \Leftrightarrow s\left(t\right)<s_{\min} \\ s_{\max} \Leftrightarrow s\left(t\right)>s_{\max} \\ s\left(t\right)\ \ldots \quad \text{otherwise} \end{cases} \qquad s_{\min}=-\frac{9}{10} \qquad s_{\max}=\frac{\pi}{2}$$

- the result is vector `sp_t`
- use functions `min` and `max` with two input parameters, see Matlab Help for details
- use the following code to check your output:

```
>> close all;
>> plot(t, s_t); hold on;
>> stem(t, sp_t, 'r');
```



Matlab Editor

- it is often wanted to evaluate certain sequence of commands repeatedly $\Rightarrow$ utilization of Matlab scripts (plain ACSII coding)
- the best option is to use `Matlab Editor`
  - to be opened using:

```
>> edit
```

  - or in Matlab < R2012a: `Start` → `Desktop Tools` → `Editor`
- a script is a sequence of statements that we have been up to now typing in the command line
  - all the statements are executed one by one on the launch of the script
  - the script operates with global data in `Matlab Workspace`
  - suitable for quick analysis and solving problems involving multiple statements
- there are specific naming conventions for scripts (and also for functions as we see later)

Script execution, m-files

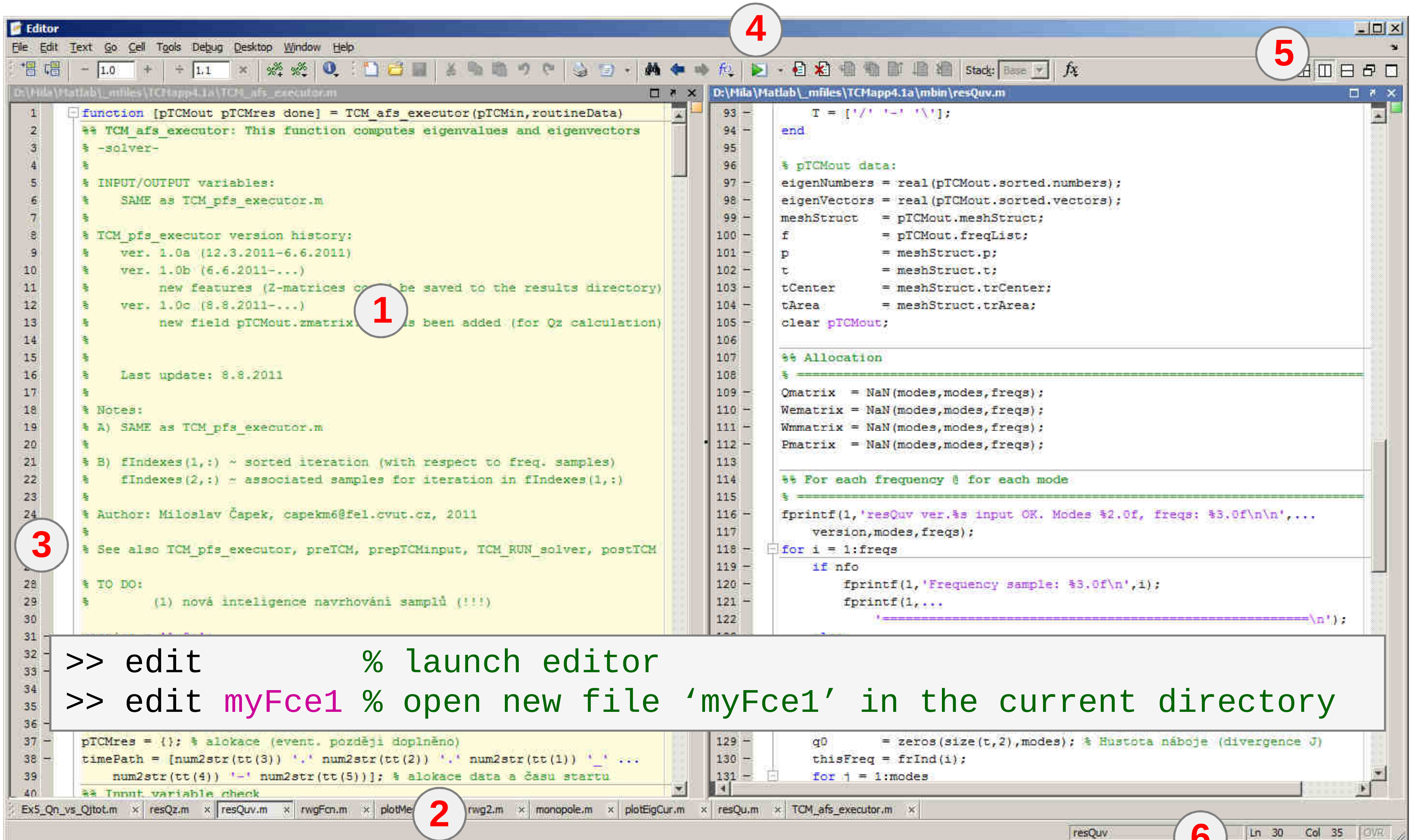
- to execute script:
  - F5 function key in Matlab Editor
  - Current Folder → select script → context menu → Run
  - Current Folder → select script → F9
  - From the command line:

```
>> script_name
```

- Scripts are stored as so called m-files
  - `.m`
  - caution: if you have Mathematica installed, the .m files may be launched by Mathematica

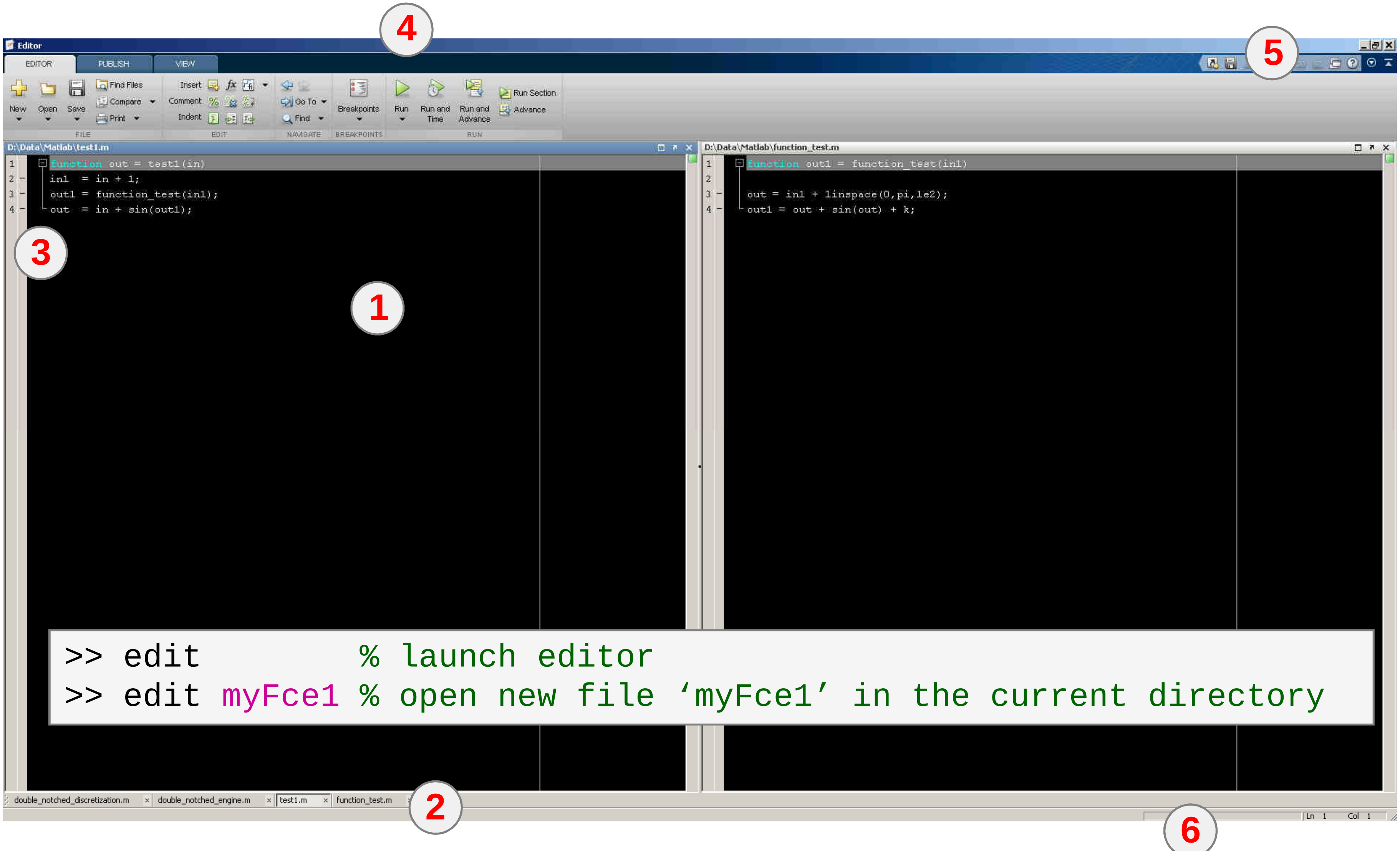
Department of Electromagnetic Field, CTU FEE, miloslav.capek@fel.cvut.cz

Matlab Editor, < R2012a




```
>> edit          % launch editor
>> edit myFce1 % open new file 'myFce1' in the current directory
```

Matlab Editor, ≥ R2012a




```
>> edit          % launch editor
>> edit myFce1 % open new file ‘myFce1’ in the current directory
```

Useful shortcuts for Matlab Editor

key	meaning
`CTRL + Pg. UP`	switch among all open m-files - one direction
`CTRL + Pg. DOWN`	- other direction
`CTRL + R`	adds '%' at the beginning of the selected lines, "comment lines"
`CTRL + T`	removes '%' from selected lines
`F5`	execute current script / function
`CTRL + S`	save current file (done automatically after pressing F5)
`CTRL + HOME`	jump to the beginning of file
`CTRL + END`	jump to the end of file
`CTRL + → / ←`	jump word-by-word or expression-by-expression to the right / left
`CTRL + W`	close current file
`CTRL + O`	activates open file dialog box (drag and drop technique also available)
`CTRL + F`	find / replace dialog box
`CTRL + G`	„go to", jumps to the indicated line number
`CTRL + D`	open m-file of the function at the cursor's position
`CTRL + I`	indention of block of lines corresponding to key words (`for` / `while`, `if` / `switch` – `case`)
`F1`	open context help related to the function at position of cursor

Matlab Editor

120 s

- open `Matlab Editor` and prepare to work with a new script, call it `signal1.m`, for instance
- use signal generation and limiting from one of the previous slides as the body of the script
- Save the script in the current (or your own) folder

- try to execute the script (F5)

```
>> edit signal1
```

```
%% script generates signal with noise
clear; clc;
t = linspace(0, 5, 5*40);
s_t = sqrt(2*pi)*sin(2*pi*t) + randn(1, 5*40);
plot(t, s_t);
```

- note: from now on, the code inside scripts will be shown without leading „>>“

Useful functions for script generation

- function `disp` displays value of a variable in `Command Window`
  - without displaying variable's name and the equation sign "="
  - Can be combined with s text (more on that later)
  - more often it is advantageous to use more complicated but robust function `sprintf`

```
>> a = 2^13-1;
b = [8*a 16*a];
b

b =

      65528      131056
```

```
a = 2^13-1;
b = [8*a 16*a];
b
```

vs.

```
a = 2^13-1;
b = [8*a 16*a];
disp(b);
```

```
>> a = 2^13-1;
b = [8*a 16*a];
disp(b);
      65528      131056
```

- function `input` is used to enter variables
  - if the function is terminated with an error, the input request is repeated

```
A = input('Enter parameter A: ');
```

  - It is possible to enter strings as well:

```
str = input('Enter String str: ', 's');
```

```
>> A = input('Enter parametr A: ');
Enter parametr A: 10.153
>> A = input('Enter string str: ', 's');
Enter string str: this is a test
>> whos
  Name      Size            Bytes  Class     Attributes

  A         1x14               28  char
  ans       1x1                 8  double
```

Matlab Editor – Exercise

600 s

- create a script to calculate compound interest*
  - the problem can be described as :

$$P = \frac{rA\left(1+\frac{r}{n}\right)^{nk}}{n\left(\left(1+\frac{r}{n}\right)^{nk}-1\right)},$$

where P is regular repayment of debt A, paid n-times per year in the course of k years with interest rate r (decimal number)

- create a new script and save it
- at the beginning delete variables and clear `Command Window`
- implement the formula first, then proceed with inputs (`input`) and outputs (`disp`)
- Try to vectorize the code, e.g. for various values of P, n or k
- Check your results (pro $A = 1000$, $n = 12$, $k = 15$, $r = 0.1$ je $P = 10.7461$)

*interest from the prior period is added to principal

Department of Electromagnetic Field, CTU FEE, miloslav.capek@fel.cvut.cz

Matlab Editor – Exercise

- try to vectorize the code, both for r and k

$$P = \frac{rA\left(1+\frac{r}{n}\right)^{nk}}{n\left(\left(1+\frac{r}{n}\right)^{nk}-1\right)}$$

- use scripts for future work with Matlab
  - bear in mind, however, that parts of the code can be debugged using command line

Linear indexing

600 s

- let's consider following matrix:

```
>> A = magic(4);
```

  - use linear indexing so that only the element with the highest value in each row of A was left (all other values set to 0); call the new matrix B

```
>> B = zeros(size(A));
>> % complete ...
```


Department of Electromagnetic Field, CTU FEE, miloslav.capek@fel.cvut.cz

Useful functions for script generation

- function `keyboard` stops execution of the code and gives control to the keyboard
  - the function is widely used for code debugging as it stops code execution at the point where doubts about the code functionality exist

```
K>>
```

  - `keyboard` status is indicated by `K>>` (K appears before the prompt)
  - The keyboard mode is terminated by `return`
- function `pause` halts code execution,
  - `pause(x)` halts code execution for `x` seconds

```
% code; code; code;
pause;
```

- see also: `echo`, `waitforbuttonpress`
  - special purpose functions

Department of Electromagnetic Field, CTU FEE, miloslav.capek@fel.cvut.cz

Matlab Editor – Exercise

360 s ↑

- modify the script for compound interest calculation in the way that
  - values *A* and *n* are entered from the command line (function `input`)
  - test the function `keyboard` (insert it right after parameter input)
    - is it possible to use `keyboard` mode to change the parameters inserted by `input`?
    - arrange for exiting the `keyboard` (`K>>`) mode, use `return`
  - interrupt the script before displaying results (function `pause`)
    - note the warning „*Paused*“ in the bottom left part of main Matlab window

Department of Electromagnetic Field, CTU FEE, miloslav.capek@fel.cvut.cz

Script commenting

- **MAKE COMMENTS!!**
  - important / complicated parts of code
  - description of functionality, ideas, change of implementation

typical comment (one-/multiple- line)

enables to separate function into more blocs (%% …)

Multiple-line comment

```
% A    = magic(3);
matX = dataIn(:,1);
SumX = sum(matX); % all members are summed
%% CELL mode (must be enabled in Editor)
disp(num2str(SumX));
Z = inv(ZZ);
%{
This is a multi-line comment.
Mostly, it is more appropriate to use more
single-line comments.
%}
```

Shortcuts:
CTRL+R
CTRL+T

Department of Electromagnetic Field, CTU FEE, miloslav.capek@fel.cvut.cz

When not making comments…

- …
  no
  one
  will
  understand!

```
edgTotal  = MeshStruct.edgTotal;
RHO_P     = zeros(3,9,edgTotal);
RHO_M     = zeros(3,9,edgTotal);
for m = 1:edgTotal
    RHO_P(:,:,m) = repmat(MeshStruct.Rho_Plus1(:,m),[1 9]);
    RHO_M(:,:,m) = repmat(MeshStruct.Rho_Minus1(:,m),[1 9]);
end
Z         = zeros(edgTotal,edgTotal) + 1j*zeros(edgTotal,edgTotal);
for p = 1:MeshStruct.trTotal
    Plus  = find(MeshStruct.TrianglePlus - p == 0);
    Minus = find(MeshStruct.TriangleMinus - p == 0);
    D     = MeshStruct.trCenter9 - ...
            repmat(MeshStruct.trCenter(:,p),[1 9 MeshStruct.trTotal]);
    R     = sqrt(sum(D.*D));
    g     = exp(-K*R)./R;
    gP    = g(:,:,MeshStruct.TrianglePlus);
    gM    = g(:,:,MeshStruct.TriangleMinus);
    Fi    = sum(gP) - sum(gM);
    ZF    = FactorFi.*reshape(Fi,edgTotal,1);
    for k = 1:length(Plus)
        n      = Plus(k);
        RP     = repmat(MeshStruct.Rho_Plus9(:,:,n),[1 1 edgTotal]);
        RPi    = repmat(MeshStruct.Rho_Minus9(:,:,n),[1 1 edgTotal]);
        A      = sum(gP.*sum(RP.*RHO_P)) + sum(gM.*sum(RP.*RHO_M));
        Z1     = FactorA.*reshape(A,edgTotal,1);
        Z(:,n) = Z(:,n) + MeshStruct.edgLength(n)*(Z1+ZF);
    end
    for k = 1:length(Minus)
        n      = Minus(k);
        RP     = repmat(MeshStruct.Rho_Minus9(:,:,n),[1 1 edgTotal]);
        RPi    = repmat(MeshStruct.Rho_Plus9(:,:,n),[1 1 edgTotal]);
        A      = sum(gP.*sum(RP.*RHO_P)) + sum(gM.*sum(RP.*RHO_M));
        Z1     = FactorA.*reshape(A,edgTotal,1);
        Z(:,n) = Z(:,n) + MeshStruct.edgLength(n)*(Z1-ZF);
    end
end
```

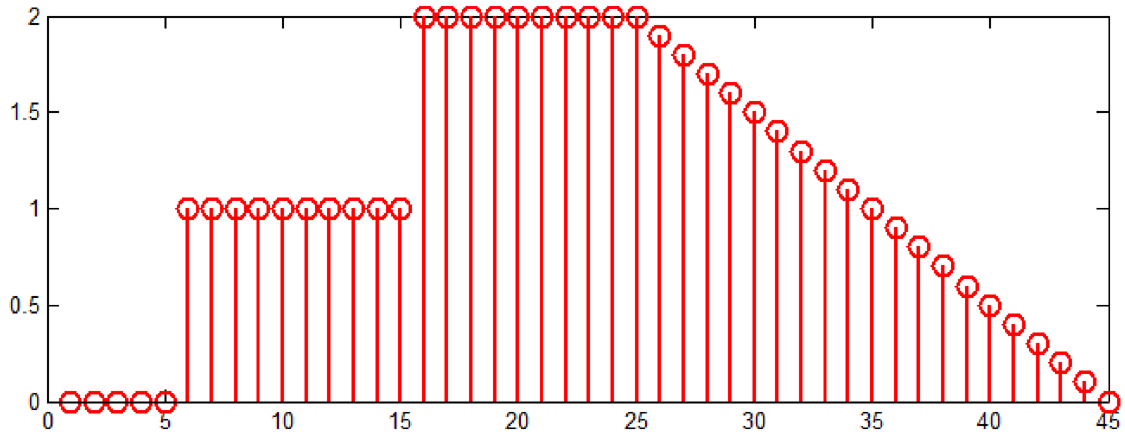
Discussed functions

`edit`	open Matlab Editor	●
`disp, pause`	display result in the command line, terminate script execution	●
`keyboard, return, input`	enables user to enter script being executed, value input request	●
`who, what, whos, which`	information on variables, files, folders	●
`cd, pwd, dir`	change directory, list folder	●
`memory, ver`	available memory information, version of Matlabu and toolboxes	●
`format, delete`	command line display format, delete file / objects	●

Exercise #1

400 s

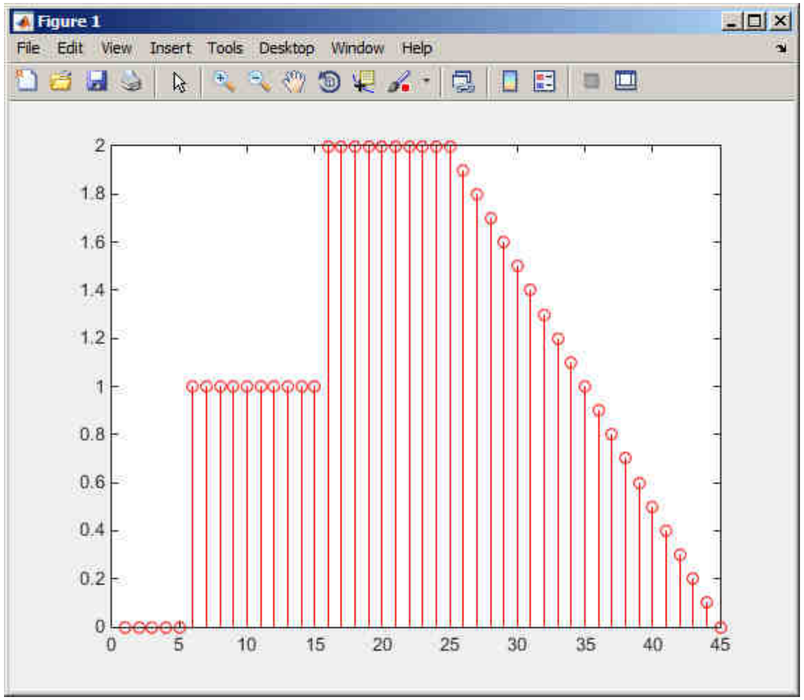
- generate vector containing following sequence



  - note the x axis (interval, number of samples)
  - split the problem into several parts to be solved separately
  - several ways how to solve the problem
  - use `stem(x)` instead of `plot(x)` for plotting
- try to generate the same signal beginning with zero …

Exercise #2

- generate vector containing following sequence



- one of possible solutions:

- or

Exercise #3

- consider following signal:

$$s(t)=\sqrt{2\pi}\sin(2\omega_0 t)+n(\mu,\sigma)$$

where the mean of normal distribution $n(\mu,\sigma)$ is $\mu=0$ (`mu`) and standard deviation $\sigma=1$ (`sigma`). Matlab syntax of n is:

```
n = mu + sigma*randn(1, N*V)
```

- create signal within time interval $\langle 5;10\rangle$ so that $N=5$ periods of the signal is depicted using $V=40$ samples per period.

- use the code in the following slide and correct errors in the code. Correct solution will be presented during next lecture.

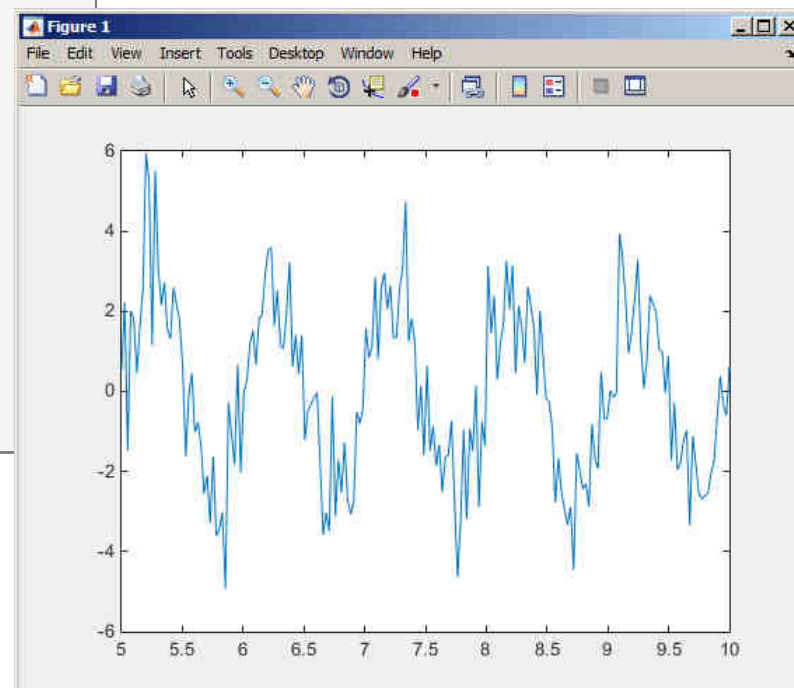
Department of Electromagnetic Field, CTU FEE, `miloslav.capek@fel.cvut.cz`

Exercise #4

400 s

```
%% TIME VECTOR GENERATION
N = 5;          % number of periods
40 = V;         % no of samples per period
k == 5;         % beginning of the interval
t = linspace[k, N+k+10), N*V];
clear;
%% NOISE VECTOR GENERATION
mu    = 2;    % mean
sigma = 0;    % standard deviation
n     = mu + sigma*randn(1, N*V);
%% NOISY SIGNAL VECTOR GENERATION
omega = pi;  % angular frequency
s_t = sqrt(2*pi)*Sin(2*omega*t)*n;
%% SIGNAL PLOTTING
plot(s_t, t)
```

- Correct solution depicts:



Exercise #5

400 s

- reflection coeff. S_{11} of a one-port device of impedance Z is given by :

$$S_{11} = 10\log_{10}\left(\left|\frac{Z - Z_0}{Z + Z_0}\right|^2\right),$$

where $Z_0 = 50\,\Omega$ and $Z = R + jX$.

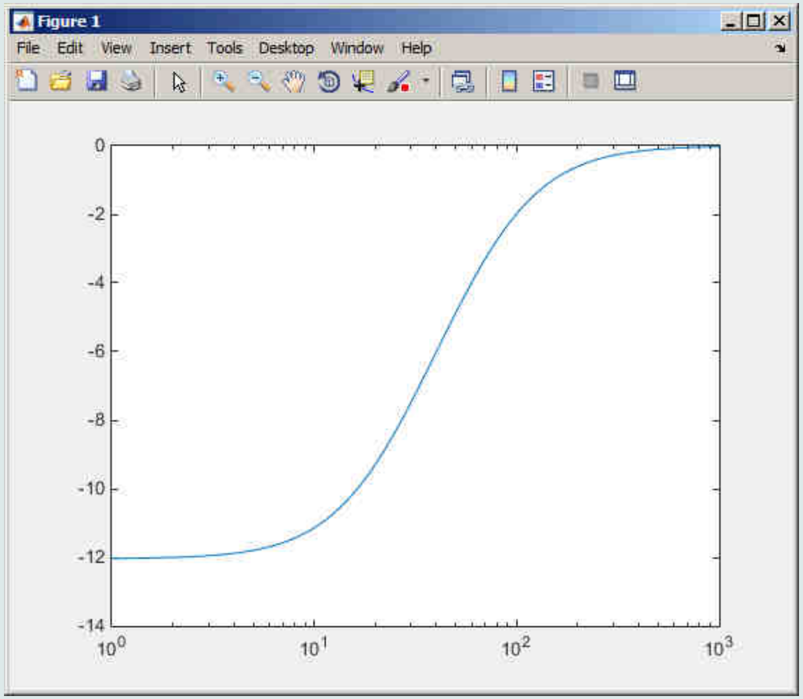
- calculate and depict the dependence of S_{11} for R = 30 Ω and X on the <1, 10^3> interval with 100 evenly spaced point in logarithmic scale

- Use the code below and correct errors in the code. Correct solution will be presented during next lecture.

```
>> 500 = Z0;                      % reference impedance
>> R   == 30;                     % real part of the impedance
>> X   = Logspace(0, 3, 1e2);     % reactance vector
>> clear;
>> Z   = i*(R + 1i*X);            % impedance
>> S11 = 10*log(abs(Z-Z0)./(Z+Z0))^2); % reflection coeff. in dB
>> semilogx(S11, X)               % plotting using log. x-axis
```

Exercise #6

- Correct solution results in the following:



Thank you!

ver. 4.2 (15/10/2015)
Miloslav Čapek, Pavel Valtr
miloslav.capek@fel.cvut.cz
Pavel.Valtr@fel.cvut.cz

Document created as part of A0B17MTB course.