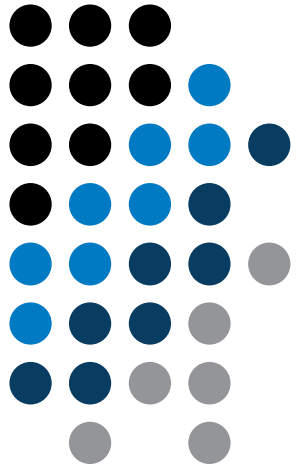
A0B17MTB – Matlab

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 …

Indexing

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

Size and type of data

Output format

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 elements 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

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

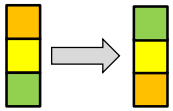
```
>> N2(1, 3)
>> N2(3, 1)
>> N2(1, end)
>> N2(end, end)
>> N2(1, :)
>> N2(1, :).'
>> N2(:, 2)
>> N2(:, 3:end)
```

```
>> 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)]
```

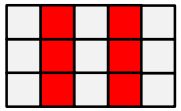
Indexing in Matlab

420 s ↑

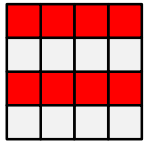
- remember the meaning of `end` and the usage of colon operator "**:**"
- try to:
  - flip the elements of the vector **N1**



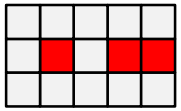
    - without using `fliplr` / `flipud` functions
  - select only the even columns of **N2**



  - 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)

Indexing in Matlab

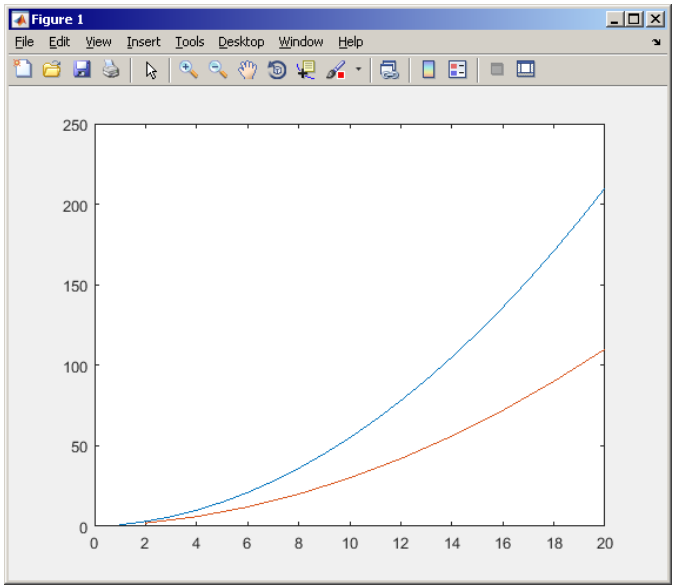
300 s ↑

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

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

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

  - 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]) = []
```

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 M)

```
>> 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.
```

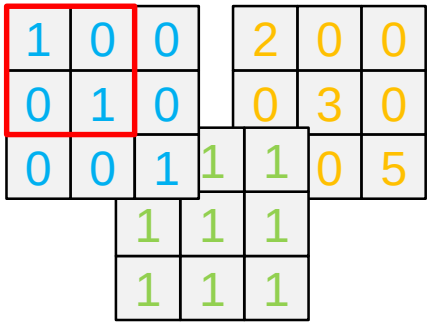
- replace the 2nd, 3rd and 5th row of matrix **A** by the first row of matrix **B**
  - assume the number of columns of matrices **A** and **B** is the same
  - consider the case where **B** has more columns than **A**
  - what happens if **B** has less columns than **A**?

Matrix creation, element replacement

300 s ↑

- create following 3D array

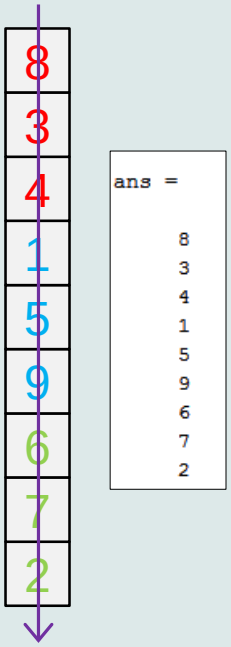
$$\mathbf{M}(:,:,1)=\begin{pmatrix}1&0&0\\0&1&0\\0&0&1\end{pmatrix},\quad \mathbf{M}(:,:,2)=\begin{pmatrix}1&1&1\\1&1&1\\1&1&1\end{pmatrix},\quad \mathbf{M}(:,:,3)=\begin{pmatrix}2&0&0\\0&3&0\\0&0&5\end{pmatrix}$$



- 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.

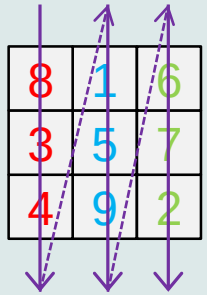



```
ans =

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

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

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

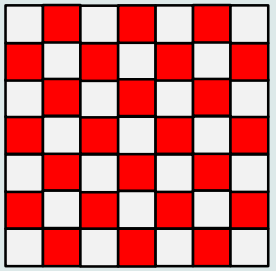
```
>> A = magic(3)

A =

     8     1     6
     3     5     7
     4     9     2

>> A(:)
```



Linear indexing - application

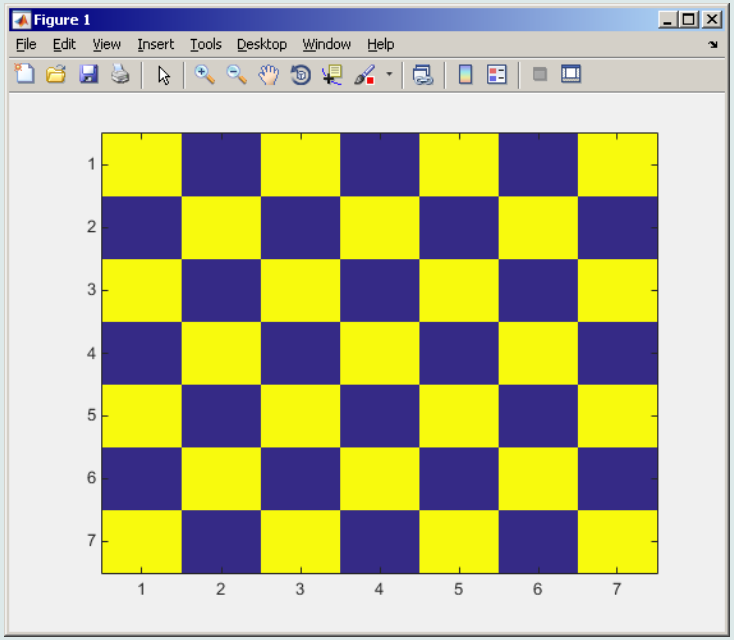
- let's consider following matrix:

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



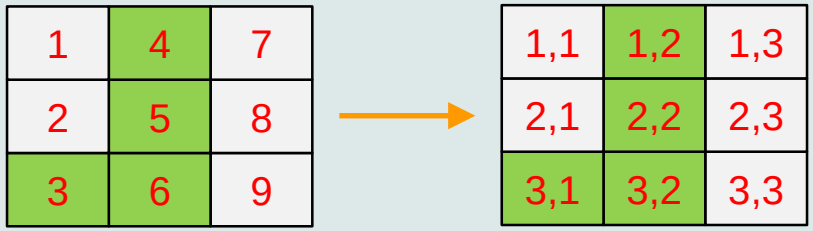
  - we set all the red-highlighted elements to zero:

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



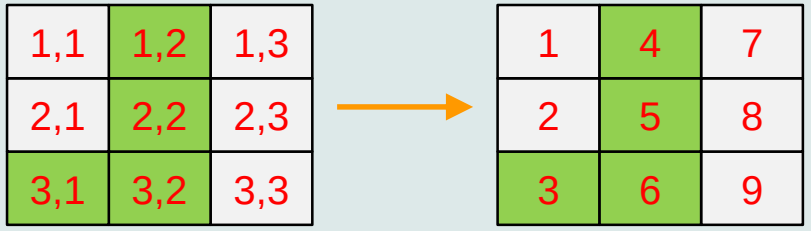
Linear indexing – `ind2sub`, `sub2ind`

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




```
>> 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




```
>> 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)
```

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;
```

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)

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

Function `prefdir`

- folder containing preferences, history, and layout files

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

- it is recommended to do not edit any file!

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 style`
  - output format does not change neither the computation accuracy nor the accuracy of stored result (`eps`, `realmax`, `realmin`, … still apply)

style	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`)

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 left division
|	ALT + 124	logical operator OR
&	ALT + 38	logical operator AND
~	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
```

Exercise #1

600 s ↑

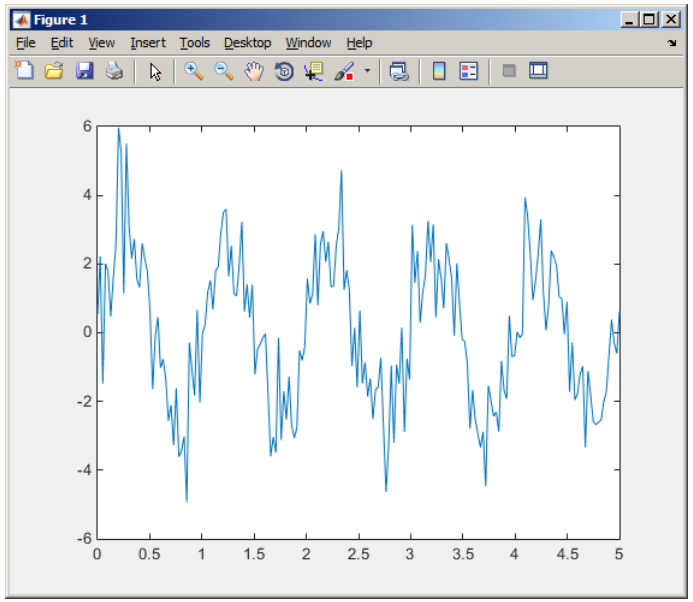
- consider signal: $s(t) = \sqrt{2\pi}\sin(2\omega_0 t) + n(\mu,\sigma), \quad \omega_0 = \pi,$
  where the mean and standard deviation of normal distribution *n* is:

  `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: $T = 1$: $t \in \left[kT, (k+N)T\right], \; k \in \mathbb{Z}^0$ (choose *k* equal for instance to 0)
  - the function $n(\mu,\sigma)$ has Matlab syntax:

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

```
>> plot(t, s_t);
```



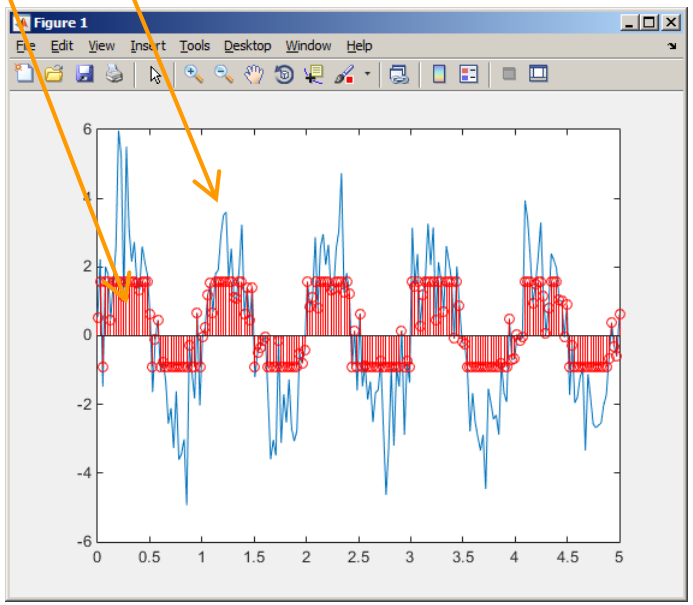
Exercise #2

600 s ↑

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

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

  - 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');
```

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 ...
```

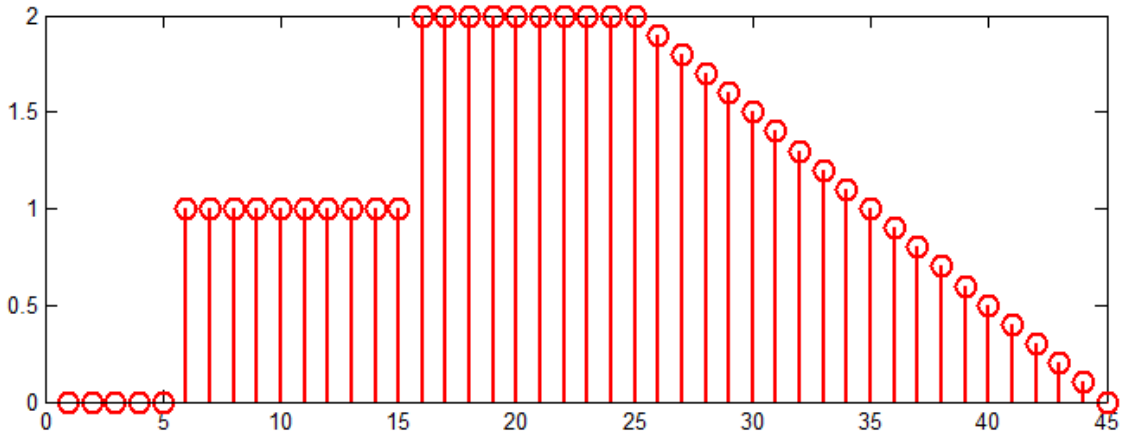
Discussed functions

`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



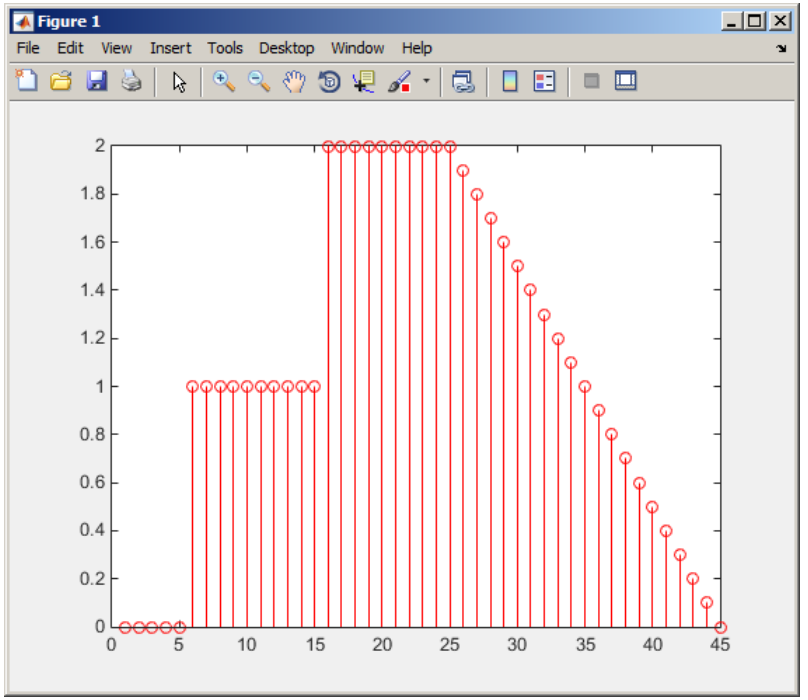
  - note the x axis (interval, number of samples)
  - split the problem into several parts to be solved separately
  - several ways how to solve the problem
  - use `stem(x)` instead of `plot(x)` for plotting
- try to generate the same signal beginning with zero …

Exercise #2

- generate vector containing following sequence

- one of possible solutions:

- or



Exercise #3

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 $Z = R + jX$.

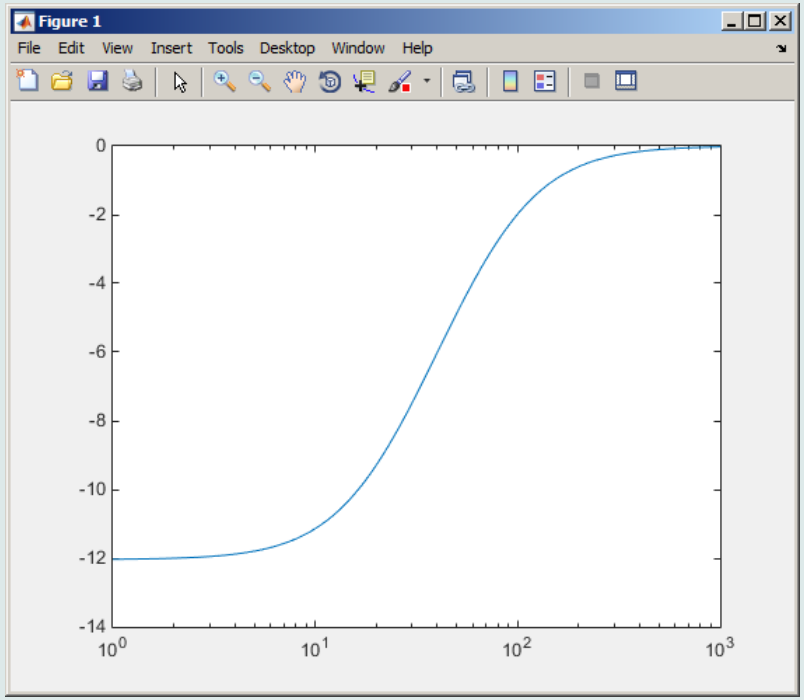
- calculate and depict the dependence of S_{11} for $R = 30\ \Omega$ and X on the $\langle 1, 10^3 \rangle$ interval with 100 evenly spaced point in logarithmic scale

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

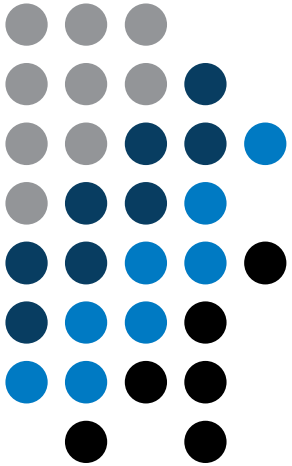
```
>> 500 = Z0;                          % reference impedance
>> R   == 30;                         % real part of the impedance
>> X   = Logspace(0, 3, 1e2);         % reactance vector
>> clear;
>> Z   = i*(R + 1i*X);                % impedance
>> S11 = 10*log(abs(Z-Z0)./(Z+Z0))^2); % reflection coeff.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

Exercise #4

- Correct solution results in the following:



Thank you!



ver. 7.1 (06/03/2017)
Miloslav Čapek, Pavel Valtr
miloslav.capek@fel.cvut.cz
Pavel.Valtr@fel.cvut.cz