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

Size and type of data

Output format

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

- 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`

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

```matlab
>> 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

- 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**

- calculate cumulative sum $S$ of a vector $x$ consisting of integers from 1 to 20
  - search Matlab help to find appropriate function (*cumulative sum*)
  
  $$x = (1 \ 2 \ \ldots \ \ 20)$$
  $$S = (1 \ 1+2 \ \ldots \ 1+2\ldots+20)$$

- 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

- 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 2\textsuperscript{nd} row of matrix A
    ```
    >> A(2, :) = []
    ```
  - remove 3\textsuperscript{rd} column of matrix A
    ```
    >> A(:, 3) = []
    ```
  - remove 1\textsuperscript{st}, 2\textsuperscript{nd} a 5\textsuperscript{th} 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 $M \times N$) by a vector $x$ (length $M$)
    
    $\text{>> } A(:, 3) = x$
  
  - 2nd, 4th a 5th row of matrix $A$ by three rows of matrix $B$ (number of columns of both $A$ and $B$ is the same)
    
    $\text{>> } A([2, 4, 5], :) = B(1:3, :)$

- 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 $B$)
    
    $\text{>> } A(2, :) = B(:, 5)$

- remember that always the size of matrices have to match!
Deleting, adding and replacing matrices

- 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

```matlab
>> 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

- create following 3D array

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

- 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.

\[ A = \text{magic}(3) \]

\[
\begin{array}{ccc}
8 & 1 & 6 \\
3 & 5 & 7 \\
4 & 9 & 2 \\
\end{array}
\]

\[
\text{ans} = \\
\begin{array}{ccc}
8 & 3 & 4 \\
1 & 5 & 9 \\
6 & 7 & 2 \\
\end{array}
\]

\[ A(:) \]
Linear indexing - application

- let’s consider following matrix:
  \[
  \text{MAT} = \text{ones}(7);
  \]

- we set all the red-highlighted elements to zero:
  \[
  \text{MAT}(2:2:end) = 0;
  \]
  \[
  \text{imagesc} \left( \text{MAT} \right);
  \]
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

- **sub2ind**: recalculates subscripts to linear index
  - 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]

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

- 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]

```matlab
>> 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

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

```
>> whos('-file', 'matlab.mat');
```
Function what, which, delete

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

  ```matlab
  >> 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

```matlab
>> 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
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

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

```
>> ver
>> V = ver

>> V = version
```
Format of command line output

- 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`)

<table>
<thead>
<tr>
<th>style</th>
<th>format description</th>
</tr>
</thead>
<tbody>
<tr>
<td>short</td>
<td>fixed 4 decimal points are displayed</td>
</tr>
<tr>
<td>long</td>
<td>15 decimal points for double accuracy, 7 decimal points for single accuracy</td>
</tr>
<tr>
<td>shortE</td>
<td>floating-point format (scientific notation)</td>
</tr>
<tr>
<td>longE</td>
<td>-/-</td>
</tr>
<tr>
<td>bank</td>
<td>Two decimal points only (euro – cents)</td>
</tr>
<tr>
<td>rat</td>
<td>Matlab attempts to display the result as a fraction</td>
</tr>
<tr>
<td>and others</td>
<td>note.: omitting <code>setting</code> parameter restores default setup</td>
</tr>
</tbody>
</table>
Format of command line output

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

```matlab
>> 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

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>[</td>
<td>ALT + 91</td>
<td>matrix definition, indexing</td>
</tr>
<tr>
<td>]</td>
<td>ALT + 93</td>
<td>-//-</td>
</tr>
<tr>
<td>{</td>
<td>ALT + 123</td>
<td>cell elements indexing</td>
</tr>
<tr>
<td>}</td>
<td>ALT + 125</td>
<td>-//-</td>
</tr>
<tr>
<td>@</td>
<td>ALT + 64</td>
<td>handle (symbolic math)</td>
</tr>
<tr>
<td>&gt;</td>
<td>ALT + 62</td>
<td>relation operator</td>
</tr>
<tr>
<td>&lt;</td>
<td>ALT + 60</td>
<td>-//-</td>
</tr>
<tr>
<td>\</td>
<td>ALT + 92</td>
<td>Matrix left division</td>
</tr>
<tr>
<td></td>
<td>ALT + 124</td>
<td>logical operator OR</td>
</tr>
<tr>
<td></td>
<td>ALT + 38</td>
<td>logical operator AND</td>
</tr>
<tr>
<td></td>
<td>ALT + 126</td>
<td>-//-</td>
</tr>
<tr>
<td></td>
<td>ALT + 94</td>
<td>power</td>
</tr>
</tbody>
</table>

- for more see: [http://www.asciitable.com/](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

```matlab
>> !calc
```

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

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

- it is possible to run Matlab with several ways

```matlab
>> 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!!!

```matlab
>> 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

• 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:

\[ \mu = 0, \quad \sigma = 1 \]

• create time dependence of the signal spanning \( N = 5 \) periods of the signal using \( V = 40 \) samples per period

• one period: \( T = 1: t \in [kT, (k + N)T], \ k \in \mathbb{Z}^0 \) (choose \( k \) equal for instance to 0)

• the function \( n(\mu, \sigma) \) has Matlab syntax:

\[
\text{>> } n = \text{mu} + \text{sigma*randn(1, N*V)}
\]

\[
\text{>> plot(t, s_t)};
\]
Exercise #2

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

\[
s_p(t) = \begin{cases} 
  s(t) & s(t) < s_{\text{min}} \\
  s_{\text{max}} & s(t) > s_{\text{max}} \\
  s(t) & \text{otherwise}
\end{cases}
\]

- 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:

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

- let’s consider following matrix:

\[
\begin{bmatrix}
\ast & \ast & \ast & \ast \\
\ast & \ast & \ast & \ast \\
\ast & \ast & \ast & \ast \\
\ast & \ast & \ast & \ast \\
\end{bmatrix}
\]

- 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

\[
\begin{align*}
& \text{let } a = \text{magic}(4) ; \\
& \text{let } B = \text{zeros(size(A))} ; \\
& \text{let } \% \text{ complete } ...
\end{align*}
\]
### Discussed functions

<table>
<thead>
<tr>
<th>Command(s)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>who, what, whos, which</td>
<td>information on variables, files, folders</td>
</tr>
<tr>
<td>cd, pwd, dir</td>
<td>change directory, list folder</td>
</tr>
<tr>
<td>memory, ver</td>
<td>available memory information, version of Matlab and toolboxes</td>
</tr>
<tr>
<td>format, delete</td>
<td>command line display format, delete file / objects</td>
</tr>
</tbody>
</table>
Exercise #1

- 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

- 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 \(<1, 10^3>\) 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.

Listing:

```matlab
>> 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.capec@fel.cvut.cz
Pavel.Valtr@fel.cvut.cz

Apart from educational purposes at CTU, this document may be reproduced, stored or transmitted only with the prior permission of the authors. Document created as part of A0B17MTB course.