EMiT
Electromagnetic Tuesday

ČVUT
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

WORKSHOPY
EXPERIMENTY
SOUTĚŽE
KATEDRA JINAK
OBČERSTVENÍ PRO VŠECHNY

27. 2. 2018 | 16:00-20:00

FEL ČVUT V PRAZE
TECHNICKÁ 2 | PRAHA - DEJVICE
BLOK B2 | 6. PATRO

elmag.org

EMiT
Electromagnetic Tuesday

ČVUT
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

WORKSHOPY | EXPERIMENTY | SOUTĚŽE

INTERAKTIVNÍ NÁVRH ANTÉNY
VAN DE GRAAFFŮV GENERÁTOR
TESLŮV TRANSFORMÁTOR
VÝROBA A MĚŘENÍ ANTÉN
RADAR PRO MĚŘENÍ RYCHLOSTI
ŠÍŘENÍ RADIOVÝCH VLN V PROSTORÁCH KATEDRY
KONEKTOROVÁNÍ OPTICKÝCH VLÁKEN
MICHELSONŮV INTERFEROMETR
APLIKACE ELEKTROMAGNETICKÉHO POLE V MEDICÍNĚ

27. 2. 2018 | 16:00-20:00

FEL ČVUT V PRAZE
TECHNICKÁ 2 | PRAHA - DEJVICE
BLOK B2 | 6. PATRO

elmag.org

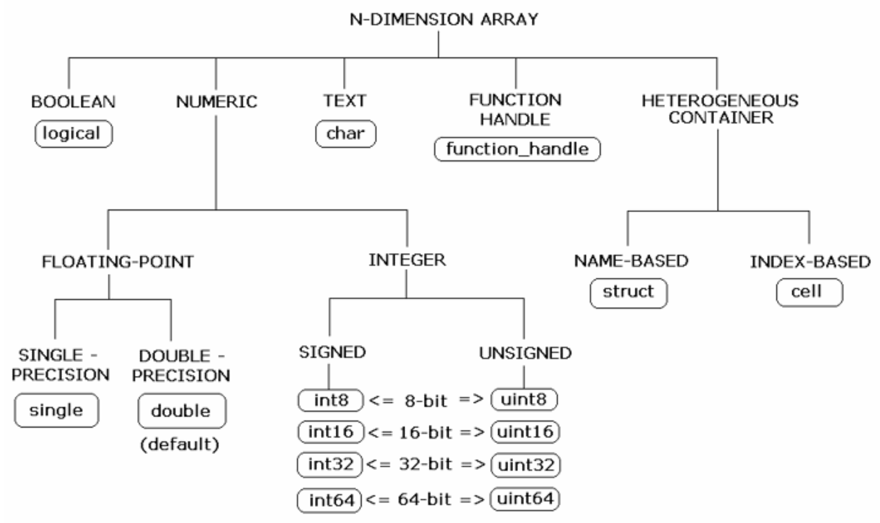
A0B17MTB – Matlab

Part #1

Miloslav Čapek
miloslav.capek@fel.cvut.cz
Viktor Adler, Pavel Valtr, Filip Kozák

Department of Electromagnetic Field
B2-634, Prague

You will learn …



Scalars, vectors, matrices (class numeric)

Matrix operations

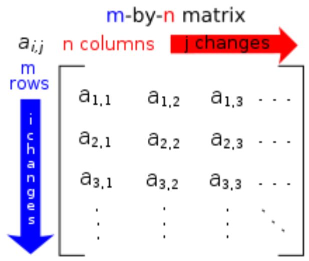
`Command Window, Command History`

Saving and loading variables

Exercises

Matrices in Matlab

- matrix is a basic data structure in Matlab
- there are following types depending on size :
  - 1×1 (scalar)
  - M×1, or 1×N (vector)
  - M×N (matrix)
  - array (multidimensional matrices) M×N×P×Q×R×…


m-by-n matrix
$a_{i,j}$
n columns
j changes
m rows
i changes
$a_{1,1}$ $a_{1,2}$ $a_{1,3}$
$a_{2,1}$ $a_{2,2}$ $a_{2,3}$
$a_{3,1}$ $a_{3,2}$ $a_{3,3}$

  - can be complex
  - can contain text as well (beware the length)

Matrix creation

- following techniques are available:
  - element-by-element entering (suitable for small matrices only)
  - colon notation „`:`“ to define elements of a series
  - generation by built-in functions
  - generation of matrices in m-files
  - import and export from/to external files (`.mat`, `.txt`, `.xls`)

Matrix construction element-by-element

- test following commands to construct matrices by element enumeration
  - suitable for small matrices only

```
>> a1 = -1
>> a2 = [-1]    % brackets are redundant
```

$$a_1 = a_2 = -1$$

$$\mathbf{v}_1 = \begin{pmatrix} -1 & 0 & 1 \end{pmatrix}$$

$$\mathbf{v}_2 = \begin{pmatrix} -1 \\ 0 \\ 1 \end{pmatrix}$$

```
>> v1 = [-1 0 1]
>> v2 = [-1; 0; 1]
```

```
>> M1 = [-1 0 1; -2 0 2]
>> M2 = [-1 -2; 0 0; 1 2]
>> M3 = [[-1 -2]; [0 0]] % inner brackets are redundant
```

$$\mathbf{M}_1 = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \end{pmatrix}, \quad \mathbf{M}_2 = \begin{pmatrix} -1 & -2 \\ 0 & 0 \\ 1 & 2 \end{pmatrix}, \quad \mathbf{M}_3 = \begin{pmatrix} -1 & -2 \\ 0 & 0 \end{pmatrix}$$

Matrix construction element-by-element

90 s ↑

- construct following matrices:
  - matrix values are defined inside square brackets `[ ]`
  - semicolon „`;`“ separates individual rows of a matrix

$$\mathbf{A} = \begin{pmatrix} -1 & 1 \\ 1 & -2 \end{pmatrix} \qquad \mathbf{B} = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

Matrix construction

120 s ↑

- semicolon placed at the end of a command suppresses display of the output in `Command Window`

```
>> a = 1
>> b = 5;
```

- when more than one command on the same line, coma is used to separate each command

```
>> a = 1, b = 5
>> a = 1; b = 5;
```

  - note: it is possible to copy and paste code including ">>"

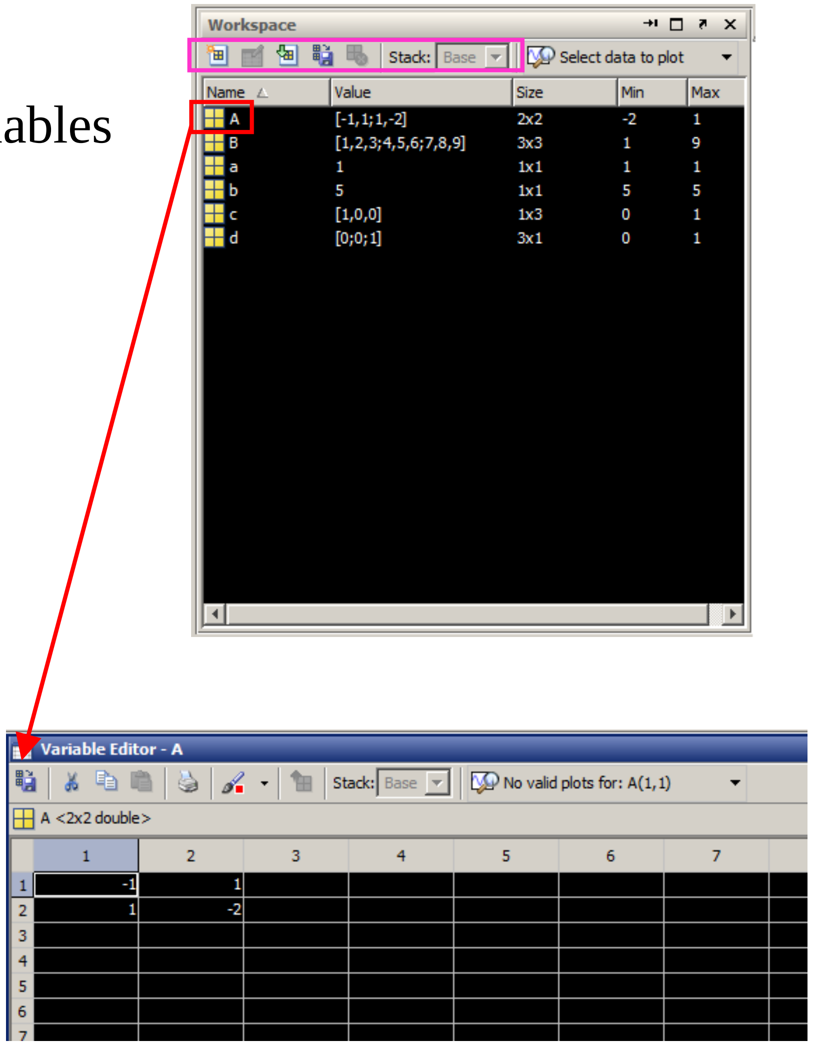
```
>> c = [1 0 0]
>> d = [0; 0; 1]
```

- "row" vs. "column" vector

$$\mathbf{c} = \begin{pmatrix} 1 & 0 & 0 \end{pmatrix} \qquad \mathbf{d} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

Workspace browser

- new variables
- deleting / modification of existing variables
- saving / loading
- size, elements of variables
  - other information can be added
- fast data plotting option
- all operations can be carried out using Matlab functions that we learn later, e.g. `min`, `max`, `size`, `length`



Name	Value	Size	Min	Max
A	[-1,1;1,-2]	2x2	-2	1
B	[1,2,3;4,5,6;7,8,9]	3x3	1	9
a	1	1x1	1	1
b	5	1x1	5	5
c	[1,0,0]	1x3	0	1
d	[0;0;1]	3x1	0	1

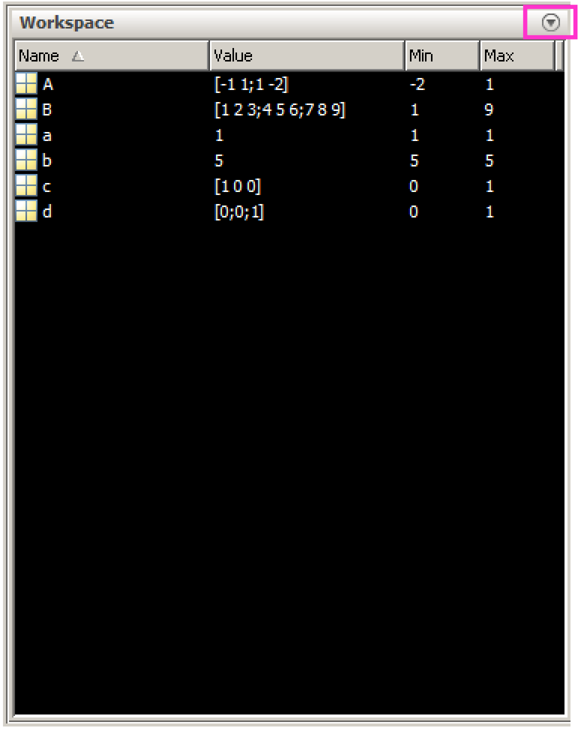
	1	2	3	4	5	6	7
1	-1	1					
2	1	-2					
3							
4							
5							
6							
7							

Workspace browser

- Workspace now contains variables `A`, `B`, `a`, `b`, `c`, `d` (from previous slides)
  - all variables in the base workspace are displayed
- variable `ans` contains the last result
  - can be used for calculation
  - overwritten by each command input!

```
>> 2*2, ans^2
```

- try to edit variables `A`, `a`
  - by a Matlab command directly
  - by change of value in `Workspace` browser
- try to delete variables `B`, `c`



Name	Value	Min	Max
A	[-1 1;1 -2]	-2	1
B	[1 2 3;4 5 6;7 8 9]	1	9
a	1	1	1
b	5	5	5
c	[1 0 0]	0	1
d	[0;0;1]	0	1

Basic math operators

- of several types:
  - arithmetic
    - matrix
    - vector
  - relational
  - logical
  - and other (to be mentioned later...)

`+`	addition
`-`	subtraction
`*`	multiplication
`^`	power
`'`	transpose
`\`	left matrix division
`/`	right matrix division
`.`	dot notation

- other operations using Matlab functions
  - complex conjugate,
  - sum, determinant, square root
  - and hundreds of other functions …

Operator Precedence in Matlab

- according to the following table
  - see Matlab → Language Fundamentals → Operators and Elementary Operations → Arithmetic

higher priority

lower priority

1	parentheses	`()`					
2	transpose, power	`'`	`.'`	`^`	`.^`		
3	unary plus, unary minus, logical negation	`+`	`-`	`~`			
4	multiplication, division	`*`	`.*`	`/`	`\`	`./`	`.\`
5	addition, subtraction	`+`	`-`				
6	colon operator	`:`					
7	relational operators	`<`	`>`	`<=`	`>=`	`==`	`~=`
8	logical AND (element-wise)	`&`					
9	logical OR (element-wise)	`|`					
10	logical AND (short-circuit)	`&&`					
11	logical OR (short-circuit)	`||`					

Basic math operators

200 s ↑

- type in following commands
  - zero can be omitted with a decimal number beginning with zero (not recommended!)

```
>> a3 = -2/4
>> a4 = -0.5
>> a5 = -.5
```

  - what is the difference between `a3`, `a4`, `a5`?
  - beware the precedence of operators (we see in the next slides):

```
>> 3*5*6
>> a1 = 15
>> a2 = 10;
>> a2/a3
>> a2/a3*a4
>> a2/(a3*a4)
```

    - explain the difference between `a2/a3*a4` and `a2/(a3*a4)`
  - verify the rules of operator precedence from the previous slide

Lengthy commands in Matlab

120 s

- it is suitable to structure command blocks for clarity:
  - next line: SHIFT+ENTER

```
>> A = [1 1 1]; B = [2 2 2]; % SHIFT+ENTER
C = [2 3 2];
```

- three dots notation
  - for continuation of the same command on the next line
  - compare results:

```
>> A = [1 1 ...
2 3]
```

```
>> A = [1 1
2 3]
```

Basic math functions

- math functions in Matlab are generally divided in three groups:
  - scalar
    - function operates over individual elements of a matrix
    - e.g.: `sin`, `sqrt`, `log`, `factorial`
  - vector
    - Function operates over individual rows/columns of a matrix
    - e.g.: `sum`, `max`
  - matrix
    - function operates over whole matrix
    - e.g.: `det`, `trace`

Basic math functions #1

600 s ↑

- using Matlab help, calculate following expression: $a\sin^2(\alpha)+a\cos^2(\alpha)-a$
  - use numerical values you choose

- verify following logarithmic identity: $\log_{10}(a)+\log_{10}(b)-\log_{10}(ab)=0$

- find sum of all elements in individual rows of the following matrix

$$\mathbf{T}=\begin{pmatrix}\frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \frac{1}{5}\\ 6 & 7 & 8 & 9\\ 0.2 & 0.3 & 0.4 & 0.5\end{pmatrix}$$

Basic math functions #2

600 s ↑

- assume following vectors **u**, **v** : $\mathbf{u} = \begin{pmatrix}1 & 2 & 3\end{pmatrix}, \quad \mathbf{v} = \begin{pmatrix}3 & 2 & 1\end{pmatrix}$
  - calculate

$$\mathbf{u}\mathbf{v}^{\mathrm{T}}, \qquad \mathbf{v}\mathbf{u}^{\mathrm{T}},$$
$$\mathbf{v}^{\mathrm{T}}\mathbf{u}, \qquad \mathbf{u}^{\mathrm{T}}\mathbf{v},$$
$$\mathbf{u}\cdot\mathbf{v}, \qquad \mathbf{u}\times\mathbf{v},$$

A

$$\begin{bmatrix}1 & 2\\ 3 & 4\\ 5 & 6\end{bmatrix}$$

`wikipedia.org`

  - following functions are needed:
    - `transpose` (`.'`) of a matrix
    - `dot` scalar product
    - `cross` product
  - what is the result of the above mentioned operations?

Basic math functions #3

420 s

- use following code and round the resulting number to:

```
>> r = 1 + 10*rand(1)
```

  - (a) nearest integer
  - (b) nearest integer greater than `r`
  - (c) nearest integer lower than `r`
  - (d) zero
  - (e) zero with precision of 2 decimal digits

note: one of the functions is called `round`

- find remainder after `r` is divided by `0.1`
  - *modulus* vs. *remainder after division*

Matrix division in Matlab

- matrix operation
- two cases are distinguished: <u>left</u> division („`\`“ - `mldivide`) and <u>right</u> division („`/`“ - `mrdivide`)
  - `A` is invertible (regular), `b` is row (column) vector

$$\mathbf{Ax} = \mathbf{b}$$

$$\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$$

solution to linear system of equations

$$\mathbf{xA} = \mathbf{b}$$

$$\mathbf{x} = \mathbf{bA}^{-1}$$

```
>> x = A \ b
```

```
>> x = b / A
```

Basic math functions #4

500 s ↑

- find the sum of diagonal elements (trace of a matrix) of the matrix **T** with elements coming from normal distribution with mean equal to 10 and standard deviation equal to 4

```
>> T = 10 + 4*randn(7, 7);
```

- find determinant of matrix **U**

$$\mathbf{U} = \begin{pmatrix} 1 & 2 & \dfrac{17}{81} \\ 0 & 2 & 0 \\ 0 & -2 & -1 \end{pmatrix}$$

- solve the linear system of equations

$$\begin{aligned} x_1 + 2x_2 + 3x_3 &= 6 \\ 4x_1 + 5x_2 + 6x_3 &= 15 \\ 7x_1 + 8x_2 + x_3 &= 16 \end{aligned}$$

$$\mathbf{A}\mathbf{x} = \mathbf{b}$$

$$\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$$

Matlab commands

- Matlab is **cAsE sEnSiTiVe**
  - almost entirely, with certain exceptions (properties of graphic objects, ...)
  - pay attention to typos and variable names (see later)
    - new versions of Matlab offer certain options

```
>> AA = [1 1 1]
>> Aa
```

  - beware of different syntax in Mathematica
    - following syntax is incorrect both in Matlab and Mathematica:

```
>> Sin(pi/2) % function names start with lower case
>> cos[pi/3] % function input is in parentheses ()
```

Predefined values in Matlab

- Matlab contains several predefined values
  - `eps` – precision of single/double numbers
    - `eps` determines the shortest distance between two single/double numbers
  - `ans` – most recent answer
  - `NaN` – *not a number* (every expression containing `NaN` results is `NaN`)
    - NaN can be used advantageously in some cases
  - `Inf` – *infinite number* (variable `Inf` can be used in calculation :))
    - pay attention to Inf propagation throughout your code (use allowed operations only)
  - `i`, `j` – complex unit
    - they are all basically functions (without input parameters)
  - check results of the following expressions:

```
>> t1 = 10/0     % t1 = Inf
>> t2 = 0/0      % t2 = NaN
>> t3 = t1*5     % t3 = Inf
>> t4 = t1 + t2  % t4 = NaN
```

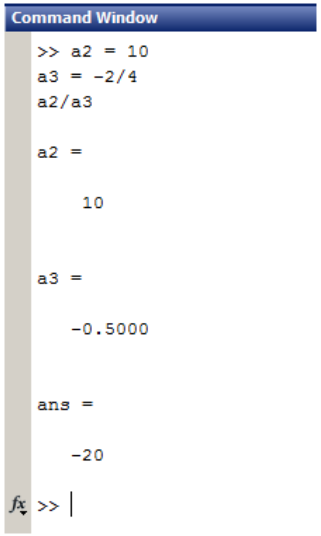
  - `pi`, `intmin`, `intmax`, `realmin`, `realmax`, ... (functions)

Workspace – output deletion #1

- to clean (erase) the command window:

```
>> home % cursor (>>) is shifted to the top-left position
>> clc  % Command Window is erased
```

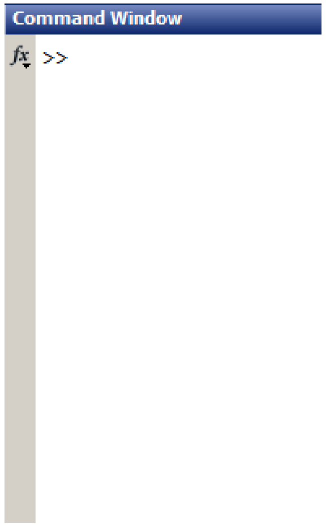
- try and compare



```
Command Window
>> a2 = 10
a3 = -2/4
a2/a3

a2 =

    10


a3 =

   -0.5000


ans =

   -20

>> |
```

```
>> home
```

```
>> clc
```



```
Command Window
>>
```

Workspace – output deletion #2

- to clean one (or more) variable, use `clear`

```
>> clear            % whole Workspace is deleted
>> clear XX         % variable XX is deleted
>> clear XX YY      % variables XX and YY are deleted
>> clear z*         % everything starting with 'z' is deleted
```

- `clear` clear has a number other options (graphics, I/O)

- try to delete selected variables in workspace

Workspace – output deletion #3

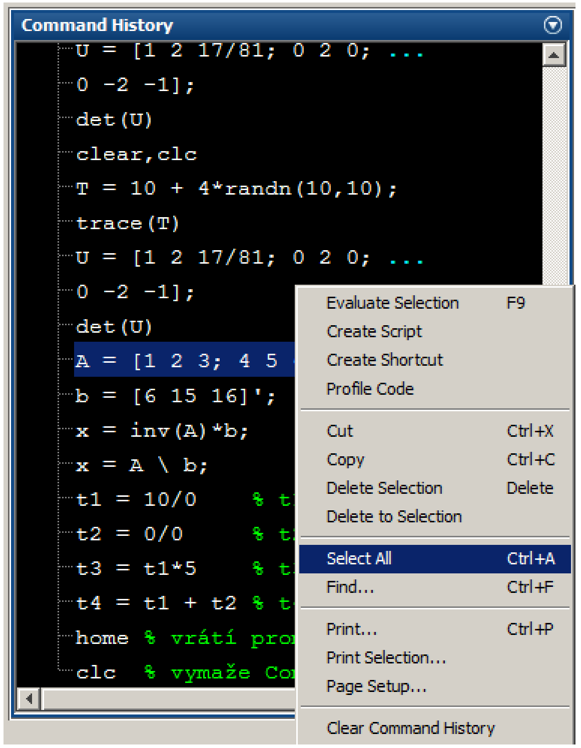
- to delete all variables except for one (or several):

```
>> clearvars -except a3 % clears everything except a3
```

- further information in `doc clear`, `doc clearvars`

Command History window

- `Command History` window stores all commands from the `Command Window`
- `Command History` accessible though (↑ or ↓)
- it is possible to filter out past commands by
  - e.g. `>> A = [` + ↑
- It is possible to copy-and-paste entire `Command History`
  - SHIFT / CTRL / CTRL+A → CTRL+C
- later on, we will work with scripts and functions to store all the commands/code



```
Command History
U = [1 2 17/81; 0 2 0; ...
0 -2 -1];
det(U)
clear,clc
T = 10 + 4*randn(10,10);
trace(T)
U = [1 2 17/81; 0 2 0; ...
0 -2 -1];
det(U)
A = [1 2 3; 4 5
b = [6 15 16]';
x = inv(A)*b;
x = A \ b;
t1 = 10/0     % t
t2 = 0/0      % t
t3 = t1*5     % t
t4 = t1 + t2 % t
home % vrátí pro
clc  % vymaže Co
```

Variables storing and loading

- existing variables in Matlab `Workspace` can be stored on disk

```
>> save % stores all variables in matlab.mat in current folder
>> save task1 % stores all variables in task1.mat
>> save task1 a b c % stores variables „a“, „b“ and „c“ in task1.mat
```

- CTRL+S in `Command Window` / `Command History`

- loading variables is analogical

```
>> load % loads matlab.mat in current folder
>> load task1 % loads all variables from task1.mat
>> load task1 a b c % loads variables „a“, „b“ and „c“ from task1.mat
```

  - alternatively, drag & drop the file from `Current Folder` in `Command Window`

Storing history and variables

180 s ↑

- save today's `Command History`
  - use `*.txt` file
- store all variables from `Workspace` in `Data.mat`
- try to store selected variables only
- clear `Workspace` and load above mentioned files
  - both storing and loading can be carried out using mouse!!

`.mat` file structure

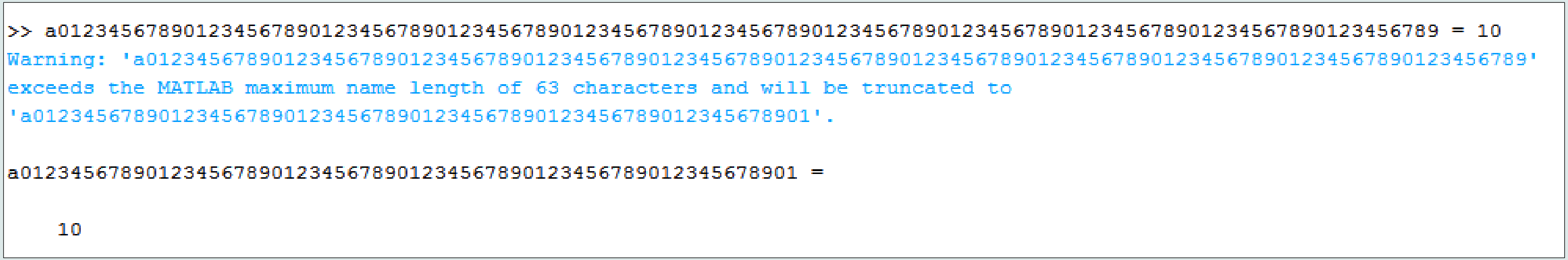
- `.mat` files of the 7.3 version have the HDF5 format
  - HDF = Hierarchical Data Format
  - enable to store variables exceeding 2GB (64-bit system)
  - scientific format for data storing
- advantages of accessing HDF directly for certain applications:
  - speed
  - it is possible to define structure of the file and the stored data
  - Matlab *High-Level* functions and HDF *Low-Level* functions are available
- for more detailed information see:
  - MATLAB → Data and File Management → Data Import and Export → Scientific Data

Variable names #1

- max. 63 characters starting with a letter (`>> namelengthmax`)
  - underscore is allowed in the variable name „_“ (not at the beginning!)
  - characters not allowed are colon „`:`“, hyphen „`-`“ and others
- lowercase letters in the names of scalars and variables (`a = 17.59;`)
- matrix names usually start with a capital letter (`A = [ ... ]`)
  - clear huge matrices after they are used (`clear ...`, memory’)
- iteration variables, variables used in `for` cycles usually named `m`, `n`, `k` etc.
  - it is advisable to avoid `i`, `j` (complex unit)
- chose the names to correspond to the purpose of the variable
- avoid, if possible, standalone letter `'l'` (to be confused with `1`) and predefined variables in Matlab environment

Variable names #2

- exceeding the maximum variable's name length :



```
>> a01234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789 = 10
Warning: 'a01234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789'
exceeds the MATLAB maximum name length of 63 characters and will be truncated to
'a01234567890123456789012345678901234567890123456789012345678901'.

a01234567890123456789012345678901234567890123456789012345678901 =


    10
```

Variable names #3

- samples of valid variable names

```
a, A, b, c, x1, x2, M_12, test1, matrix_A, fx, fX
```

- samples of invalid variable names

```
1var      % starts with a number (not possible in Matlabu)
matrix A % contains space
coef.a    % possible only if coef is of type 'struct'
Test-1    % algebraic expression: ans = Test - 1
f(y)      % makes sense when using symbolic expressions
```

- samples of valid numbers in Matlab
  - pay attention to miss inserted spaces after exponent and imaginary unit

```
3, -66, +0.0015, .015, 1.6025e-10, 3i, 3.17e5i, -3.51j
```

Discussed functions

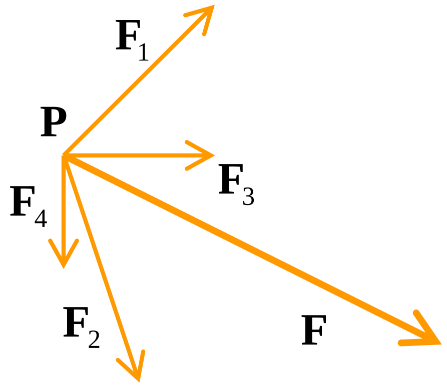
`sin, cos`	trigonometric functions	
`sqrt`	square root	
`max`	largest element of column of a matrix; largest element of a vector	●
`sum`	sum of elements of column of a matrix; sum of elements of a vector	●
`log, log10`	natural logarithm, logarithm with base 10	
`factorial`	factorial	
`det, trace`	determinant of a (square) matrix, trace of a (square) matrix	
`transpose`	transpose	
`dot, cross`	scalar product, vector product	●
`inv`	invers of a matrix	
`round, ceil, floor, fix`	rounding	
`rem`	remainder after division	
`rand, randn`	random number generation	
`save, load`	storing, loading of variables	●
`clear, clearvars`	deleting variables and functions, deleting variables only	●
`home, clc`	command prompt shift, clears output	
`ans, eps`	returns last answer, numerical accuracy of Matlab	●

Exercise #1

180 s

- forces were localized at point **P** in $(x-y)$ plane:

$$\mathbf{F}_1 = \begin{pmatrix} 2 & 2 \end{pmatrix} \qquad \mathbf{F}_3 = \begin{pmatrix} 2 & 0 \end{pmatrix}$$
$$\mathbf{F}_2 = \begin{pmatrix} 1 & -3 \end{pmatrix} \qquad \mathbf{F}_4 = \begin{pmatrix} 0 & -1.5 \end{pmatrix}$$



  - what is the direction of the resultant force **F**?

- normalize the resulting vector

$$\mathbf{n}_\mathrm{F} = \frac{\mathbf{F}}{|\mathbf{F}|} = \frac{\mathbf{F}}{\sqrt{F_x^2 + F_y^2 + F_z^2}}$$

Exercise #2

240 s

- type-in following commands:

```
>> clear, clc;
>> w1 = [1 2 3 4], w2 = [-2 -3 -4]
>> w3 = [-2; -3; -4]
>> w4 = w2 - w3, w5 = w2 - w1
```

  - compare differences
  - the error of calculating `w5` resides in what?

- try also

```
>> w1*3, w1 - 3,
>> w1 + [5 5 5 5],
>> w6 = 5*w1 - [3 5 6] - w2
```

- calculate the norm (magnitude) of vector `w1`
  - try more options

$$\hat{\mathbf{w}}_1 = \frac{\mathbf{w}_1}{|\mathbf{w}_1|}$$

  - how to modify the calculation in the case of a complex vector?

Exercise #3

180 s

- calculate roots of the quadratic function $-2x^2 - 5x = 3$
  - rearrange the terms of the function first

$$2x^2 + 5x + 3 = 0, \quad \Rightarrow \quad a = 2,\ b = 5,\ c = 3$$

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} = \frac{-5 \pm \sqrt{25 - 24}}{4}$$

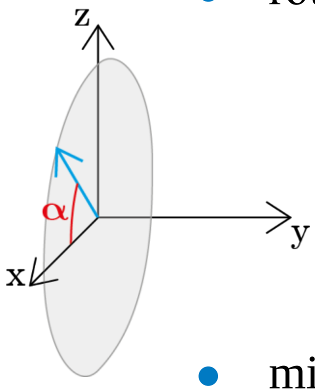
$$x_1 = -1, \quad x_2 = -\frac{3}{2}$$

  - Matlab provides particular function for calculation of roots of a function, try to search it out

Exercise #4

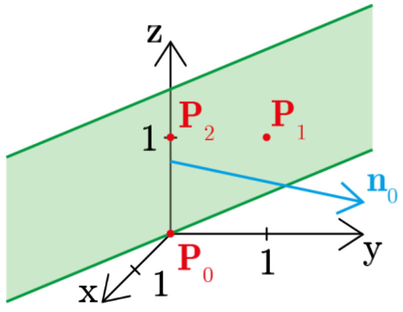
300 s ↑

- consider matrices (prepare matrices for later use)
  - rotating by angle α in *x*-*z* plane



$$\mathbf{R} = \begin{pmatrix} \cos(\alpha) & 0 & -\sin(\alpha) \\ 0 & 1 & 0 \\ \sin(\alpha) & 0 & \cos(\alpha) \end{pmatrix}$$

  - mirroring across plane $1x + 2y + 0z = 0$
    - use Householder's transform $\mathbf{P} = \mathbf{I} - 2\mathbf{n}_0\mathbf{n}_0^{\mathrm{T}}$



$$\mathbf{n}_0 = \frac{\mathbf{v}_1 \times \mathbf{v}_2}{|\mathbf{v}_1 \times \mathbf{v}_2|} \qquad \mathbf{P}_1 = [-2;\ 1;\ 0] \qquad \mathbf{P}_2 = [0;\ 0;\ 1]$$

$$\mathbf{v}_k = (\mathbf{P}_k - \mathbf{0})\begin{pmatrix} \mathbf{x}_0 \\ \mathbf{y}_0 \\ \mathbf{z}_0 \end{pmatrix}, \quad k \in \{1, 2\}$$

Exercise #5

180 s

- use rotation matrix **R** to rotate vector **k** = [1; 0; 0] by angle α = π/2

$$\mathbf{m} = \mathbf{R}\mathbf{k} = \begin{pmatrix} 0 & 0 & 1 \end{pmatrix}^{\mathrm{T}}$$

- use reflection matrix **P** across plane: $1x + 2y + 0z = 0$
  - to mirror vectors:

$$\mathbf{u}_1 = \mathbf{n}_0, \quad \mathbf{u}_2 = \begin{pmatrix} \frac{5}{2} & 0 & 3 \end{pmatrix}^{\mathrm{T}}$$

$$\mathbf{m}_1 = \mathbf{P}\mathbf{u}_1 = -\mathbf{n}_0, \quad \mathbf{m}_2 = \mathbf{P}\mathbf{u}_2 = \begin{pmatrix} \frac{3}{2} & -2 & 3 \end{pmatrix}^{\mathrm{T}}$$

- calculate the determinant of matrices **R** and **P**
  - can you interpret the results?

Thank you!

ver. 9.1 (18/02/2018)
Miloslav Čapek, Pavel Valtr
miloslav.capek@fel.cvut.cz
pavel.valtr@fel.cvut.cz