

Lecture 2: Complex Numbers, Editor, Matrix Creation

A8B17CAS

Miloslav Čapek

Department of Electromagnetic Field
Czech Technical University in Prague
Czech Republic
miloslav.capek@fel.cvut.cz

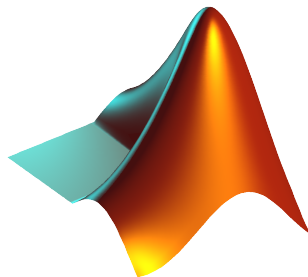
September 26
Winter semester 2025/26



Outline



1. Complex Numbers
2. MATLAB Editor
3. System of Linear Equations
4. Vector and Matrix Creation



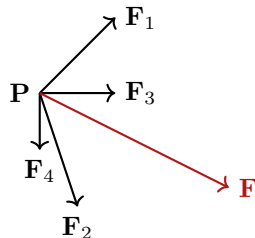
Warm Up: Adding Forces (Superposition)



- ▶ Following forces were localized at point **P** in xy plane:

$$\begin{aligned}\mathbf{F}_1 &= [2, 2] & \mathbf{F}_3 &= [2, 0] \\ \mathbf{F}_2 &= [1, -3] & \mathbf{F}_4 &= [2, -1.5]\end{aligned}$$

- ▶ What is the direction of the resultant force **F**?
- ▶ Normalize the resulting vector.



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

Bonus: Visualization of the Forces and Their Superposition

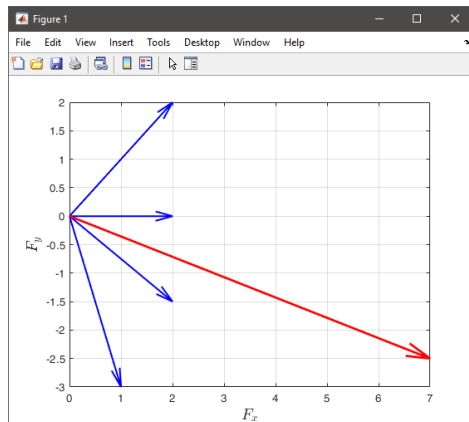


```
clear;
clc;

% User input
Fs = [2 2; 1 -3; 2 0; 2 -1.5];
F = sum(Fs, 1) % resulting force

%% Graphical output
Z = zeros(size(Fs, 1), 1);

figure('color', 'w');
quiver(Z, Z, Fs(:,1), Fs(:,2), 0, ...
    'LineWidth', 1.5, 'Color', 'b');
hold on; % allows to have more graphs in a figure
quiver(0, 0, F(1), F(2), 0, ...
    'LineWidth', 2, 'Color', 'r');
grid on; % enable grid
% add labels (LaTeX interpreter possible)
option = {'Interpreter', 'LaTeX', 'FontSize', 14};
xlabel('$F_x$', option{:});
ylabel('$F_y$', option{:});
```





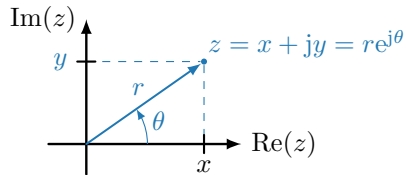
Complex Numbers I.

- Cartesian complex plane (real and imaginary parts)

$$z = x + jy$$

- Polar complex plane (modulus and argument)

$$z = |z|e^{j\phi}$$



```
>> C1 = 3 + 2j % preferred
>> C2 = 3 + 2i % preferred
>> C3 = 3 + 2*i % slow
>> C4 = 3 + 2*sqrt(-1)
>> C5 = complex(3, 2)
```



Complex Numbers II.

$$z = x + jy = |z| e^{j\phi}$$

► Frequently used functions:

x, y	<code>real, imag</code>	real and imaginary parts of a complex number
z^*	<code>conj</code>	complex conjugate
$ z $	<code>abs</code>	absolute value of a complex number
ϕ	<code>angle</code>	angle in complex plane [rad]
$x, y \rightarrow z$	<code>complex</code>	constructs complex number from real and imaginary parts
	<code>isreal</code>	checks if the input is a complex number (more on that later)
j	<code>i, j</code>	complex unit
	<code>cplxpair</code>	sort complex numbers into complex conjugate pairs



Complex Numbers — Example

- Create complex number $z = 1 + 1j$ and its complex conjugate $s = z^*$.
- Switch between Cartesian and polar form (find $|z|$ and φ).

$$z = \operatorname{Re}\{z\} + j\operatorname{Im}\{z\} = a + jb$$

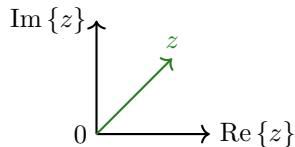
$$z = |z| e^{j\varphi}, |z| = \sqrt{a^2 + b^2}$$

$$z = |z| (\cos \varphi + j \sin \varphi)$$

- Verify Moivre's theorem:

$$z^n = (|z| e^{j\varphi})^n$$

$$z^n = |z|^n (\cos(n\varphi) + j \sin(n\varphi))$$



MATLAB Editor



```

4  % WHY(H) provides the N-th answer.
5  % Please embellish or modify this function to suit your own tastes.
6
7  % Copyright 1984-2014 The MathWorks, Inc.
8
9  if nargin > 0
10     dfilt = rng(n,'v5uniform');
11 end
12 switch randi(10)
13     case 1
14         a = special_case;
15     case {2, 3, 4}
16         a = phrase;
17     otherwise
18         a = sentence;
19 end
20 a(i) = upper(a(1));
21 disp(a);
22 if nargin > 0
23     rng(dfilt);
24 end
25
26
27 % -----
28
29 function a = special_case
30     switch randi(12)
31         case 1
32             a = 'why not?';
33         case 2
34             a = 'don't ask!';
35         case 3
36             a = 'it's your karma.';
37         case 4
38             a = 'stupid question!';
39         case 5
40             a = 'how should I know?';
41         case 6
42             a = 'can you rephrase that?';
43         case 7
44             a = 'it should be obvious.';
45         case 8
46             a = 'the devil made me do it.';
47         case 9
48             a = 'the computer did it.';
49         case 10
50             a = 'the customer is always right.';

```



Script Execution, m-files

- ▶ To execute a 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.



Useful Shortcuts

F5	run a script
F9	run selected code
%%	add cell
CTRL+SHIFT	run actual cell of code
CTRL+R	comment selected code
CTRL+T	uncomment selected code



Script Commenting

► MAKE COMMENTS!!

- Important/complicated parts of code.
- Description of functionality, ideas, change of implementation.

► Typical single-line comment:

```
% create matrix, sum all members  
matX = [1, 2, 3, 4, 5];  
sumX = sum(matX); % sum of matrix
```

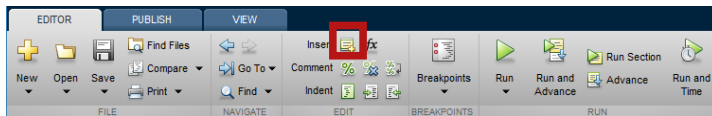
► Cell mode enables to separate script into more blocks:

```
matX = [1, 2, 3, 4, 5];  
%% CELL mode (must be enabled in Editor)  
sumX = sum(matX);
```



Cell Mode in MATLAB Editor

- ▶ Cells enable to separate the code into smaller, logically compacted parts.
 - ▶ Separator `%%`.
 - ▶ The separation is visual only, but it is possible to execute a single cell: shortcuts **CTRL+ENTER** and **CTRL+SHIFT+ENTER**.





Solving System of Linear Equations in MATLAB

- ▶ Two cases are distinguished:
 - ▶ **left** division (`\` – `mldivide`),
 - ▶ **right** division (`/` – `mrdivide`).
- ▶ Solution of a linear system of equations:
 - ▶ **A** is an invertible (regular) matrix,
 - ▶ **b** is a column (row) vector.

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

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

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

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

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

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



Linear Equations

- Find the sum of diagonal elements (trace of a matrix) of the matrix $\mathbf{T}$ with elements coming from normal distribution with mean equal to 10 and standard deviation equal to 4.
- Find determinant of matrix $\mathbf{U}$.

$$\mathbf{U} = \begin{bmatrix} 1 & 2 & 3 \\ 0 & 2 & 0 \\ 0 & -2 & -1 \end{bmatrix}$$

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

```
>> U = [1 2 3; 0 2 0; ...  
0 -2 -1];
```

- Solve the linear system of equations:

$$x_1 + 2x_2 + 3x_3 = 6$$

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

$$4x_1 + 5x_2 + 6x_3 = 15$$

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

$$7x_1 + 8x_2 + x_3 = 16$$



Predefined Values in MATLAB

- ▶ MATLAB contains several predefined values:
 - ▶ `eps` – precision of single/double numbers (Determines the shortest distance between two single/double numbers).
 - ▶ `ans` – *answer* – most recent answer.
 - ▶ `NaN` – *not a number* (every expression containing NaN 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 parameter).
 - ▶ 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
```



Format of Command Line Output

MATLAB offers number of other formatting options

- Use `>> format style`.
- Output format does not change neither the computation accuracy (single/double) nor the accuracy of stored results (eps, realmax, realmin, ...still apply).

style	format description
short	fixed 4 decimal points are displayed
long	15 decimal points for double precision, 7 decimal points for single precision
shortE	floating-point format (scientific notation)
longE	—//—
compact	suppressed the display of blank lines
and others	check <code>>> doc format</code>

- Omitting `style` parameter restores default setup!



Format of Command Line Output

- Try following output format settings:
 - Each format is suitable for different type of problems.

```
>> clc;  
>> s = [-5, 1/2, 1/3, 10*pi, sqrt(2), cos(pi/2)];  
>> format compact  
>> format long; s  
>> format longE; s  
>> format short; s  
>> format shortE; s  
>> format +; s  
>> format; s
```

- Later, we will learn how to use formatted conversion into strings (commands `sprintf` and `fprintf`).



Entering Matrices Using “:” (Colon) Operator and linspace

Large vectors and matrices require automated input.

- For equidistantly spaced values from a to b with an increment x :

```
A = a:x:b
```

- b doesn't have to be an element of the series.
- increment x can be negative.

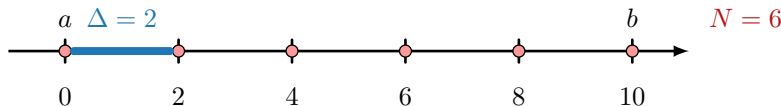
- For N equidistantly spaced values from a to b :

```
A = linspace(a, b, N)
```

```
>> A = 1:4:13
A =
    1    5    9   13
```

```
>> A = 10:-4:1
A =
   10    6    2
```

```
>> A = linspace(0, 20, 5)
A =
    0    5   10   15   20
```





Entering Matrices I.

- ▶ Using the colon operator “:” create
 - ▶ the following vector

$$\mathbf{v} = [25 \ 20 \ \dots \ -5]^T$$

- ▶ the following matrix
 - ▶ Caution, the third column can't be created using colon operator “:” only.

$$\mathbf{T} = \begin{bmatrix} -4 & 1 & \frac{\pi}{2} \\ -5 & 2 & \frac{\pi}{4} \\ -6 & 3 & \frac{\pi}{6} \end{bmatrix}$$



Entering Matrices II.

- ▶ Create a vector of 100 evenly spaced points in the interval $[-1.15, 75.4]$.
- ▶ Create a vector of 201 evenly spaced points in the interval $[-100, 100]$ sorted in descending order.
- ▶ Create a vector with spacing of -10 in the interval $[100, -100]$ sorted in descending order.
 - ▶ Try both options using `linspace` and colon `:`.



Entering Matrices Using Functions I.

- ▶ Special types of matrices of given sizes are needed quite often.
 - ▶ MATLAB offers a number of functions to serve the purpose., *e.g.*, zeros, ones, NaN, inf, eye, rand, randn, randi, true, false.
- ▶ Example: matrix filled with zeros
 - ▶ Will be used frequently.

```
zeros(m)           % matrix of size [m x m]
zeros(m, n)        % matrix of size [m x n]
zeros(m, n, p, ..) % matrix of size [m x n x p x ..]
zeros([m, n])      % matrix of size [m x n]

% see documentation for other options
```



Entering Matrices Using Functions III.

- Create following matrices
 - use MATLAB functions
 - begin with matrices you find easy to cope with.

$$\mathbf{M}_1 = \begin{bmatrix} \text{NaN} & \text{NaN} \\ \text{NaN} & \text{NaN} \end{bmatrix}$$

$$\mathbf{M}_2 = \begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix}$$

$$\mathbf{M}_3 = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & -5 \end{bmatrix}$$

$$\mathbf{M}_4 = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$



Entering Matrices

- ▶ Quite often, there are several options how to create a given matrix.
 - ▶ It is possible to use an **output of one function as an input of another** function in MATLAB:

- ▶ Consider:

```
plot(diag(randn(10, 1), 1))
```

- ▶ clarity,
- ▶ simplicity,
- ▶ speed,
- ▶ convention.

- ▶ *e.g.* band matrix with “1” on main diagonal and with “2” and “3” on secondary diagonals.

```
N = 10;  
A = diag(ones(N, 1)) + diag(2 * ones(N - 1, 1), 1) + diag(3 * ones(N - 1, 1), -1)
```

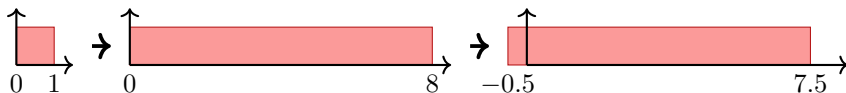
- ▶ Can be done using **for** cycle as well (see later in the semester).
- ▶ Some other idea?



Generation of a Random Matrix

- Create matrix $\mathbf{M}$ of size $\text{size}(\mathbf{M}) = [3 \ 4 \ 2]$ containing random numbers coming from uniform distribution on the interval $[-0.5, 7.5]$.

$$I(x) = (I_{\max} - I_{\min}) \text{rand}(\dots) + I_{\min}$$





Transpose and Matrix Conjugate

- Pay attention to situations where the matrix is complex, $\mathbf{A} \in \mathbb{C}^{M \times N}$.
- There are two operations:

transpose	$\mathbf{A}^T = [A_{ij}]^T = [A_{ji}]$	transpose(A)	A.'
transpose + conjugate	$\mathbf{A}^H = [A_{ij}]^H = [\mathbf{A}^*]^T$	ctranspose(A)	A'

```
>> A = magic(2) + 1j * magic(2) '
A =
  1.0000 + 1.0000i    3.0000 + 4.0000i
  4.0000 + 3.0000i    2.0000 + 2.0000i
```

```
>> A.'
ans =
  1.0000 + 1.0000i    4.0000 + 3.0000i
  3.0000 + 4.0000i    2.0000 + 2.0000i
```

```
>> A'
ans =
  1.0000 - 1.0000i    4.0000 - 3.0000i
  3.0000 - 4.0000i    2.0000 - 2.0000i
```



Matrix Operations I.

There are other useful functions apart from `transpose` (`transpose`), *e.g.*,

- ▶ vector to a diagonal matrix, matrix diagonal to a vector (`diag`),
- ▶ upper/lower triangular matrix (`triu`, `tril`),
- ▶ array replication (`repmat`),
- ▶ array reshape (`reshape`),
- ▶ array flip (`flip`),
- ▶ array rotation (`rot90`),
- ▶ circular shift (`circshift`),
- ▶ block-diagonal matrix from individual matrices (`blkdiag`),
- ▶ arranging two (or more) matrices side by side (`cat`),
- ▶ and many others...

Always check the documentation (» `doc ...`).



Matrix Operations II.

- Convert matrix $\mathbf{A}$ into the form of matrices $\mathbf{A}_1$ to $\mathbf{A}_4$.

```
A = [1 pi; exp(1) -1i]
```

$$\mathbf{A} = \begin{bmatrix} 1 & \pi \\ e & -i \end{bmatrix}$$

- Use `repmat`, `reshape`, `triu`, `tril` and `conj`.

$$\mathbf{A}_1 = \begin{bmatrix} 1 & \pi & 1 & \pi & 1 & \pi \\ e & -i & e & -i & e & -i \end{bmatrix}$$

$$\mathbf{A}_2 = \begin{bmatrix} 1 & \pi & e & -i \end{bmatrix}$$

$$\mathbf{A}_3 = \begin{bmatrix} 1 & \pi \\ e & +i \\ 1 & \pi \\ e & +i \\ 1 & \pi \\ e & +i \end{bmatrix}$$

$$\mathbf{A}_4 = \begin{bmatrix} 1 & \pi & 0 & 0 & 0 & 0 \\ e & -i & e & 0 & 0 & 0 \\ 0 & \pi & 1 & \pi & 0 & 0 \\ 0 & 0 & e & -i & e & 0 \\ 0 & 0 & 0 & \pi & 1 & \pi \\ 0 & 0 & 0 & 0 & e & -i \end{bmatrix}$$

Exercise

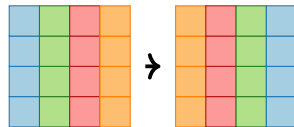


Matrix Operations III.

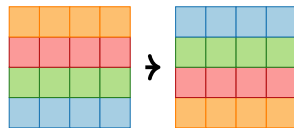
- Create the following matrix (use advanced techniques)

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 3 & 1 & 2 & 3 \\ 0 & 2 & 4 & 0 & 2 & 4 \\ 0 & 0 & 5 & 0 & 0 & 5 \end{bmatrix}$$

- Create matrix **B** by swapping columns in matrix **A**.



- Create matrix **C** by swapping rows in matrix **B**.





Matrix Operations IV. – Tensor Products

Kronecker tensor product

$$K = \text{kron}(A, B)$$

- Convolution kernel A is applied to a mask B.

Example:

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \otimes \frac{1}{2} \begin{bmatrix} 1 & -1 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 0 & 0 & 1 & -1 \\ 1 & -1 & 0 & 0 \end{bmatrix}$$

$$\text{kron}([0 \ 1; 1 \ 0], [1/2, -1/2])$$

Tensor product

$$C = \text{tensorprod}(A, B, \text{dimA}, \text{dimB})$$

- Inner product

$$\sum_n \cdots \sum_k \sum_j A_{jk \cdots n} B_{jk \cdots n} = c$$

- Outer product

$$[A_{jk \cdots n}][B_{pq \cdots t}] = [C_{jk \cdots npq \cdots t}]$$

- Tensor product

$$\sum_j \cdots \sum_p \sum_j A_{jk \cdots n} B_{pq \cdots t} = [C_{k \cdots npq \cdots t}]$$



Size of Matrices and Other Structures I.

- ▶ It is often needed to know sizes of matrices and arrays.
- ▶ Function `size` returns vector giving the size of a matrix/array.

```
A = randn(3, 5);  
d = size(A) % d = [3 5]
```

- ▶ Function `length` returns largest dimension of an array.

```
length(A) = max(size(A))
```

```
A = randn(3, 5, 8);  
e = length(A) % e = 8
```

- ▶ Function `ndims` returns number of dimensions of a matrix/array.

```
ndims(A) = length(size(A))
```

```
m = ndims(A) % m = 3
```

- ▶ Function `numel` returns number of elements of a matrix/array.

```
numel(A) = prod(size(A))
```

```
n = numel(A) % n = 120
```

- ▶ Functions `height` and `width` return number of rows and columns, respectively.



Size of Matrices and Other Structures II.

- ▶ Create an arbitrary 3D array.
 - ▶ You can make use of the following commands:

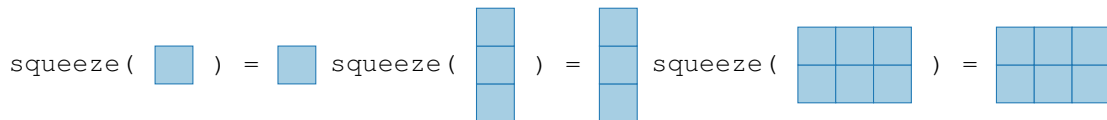
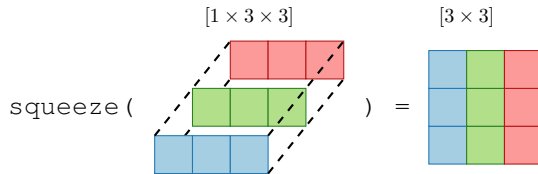
```
A = rand(2 + randi(10), 3 + randi(5));  
A = cat(3, A, rot90(A, 2))
```

- ▶ And now:
 - ▶ Find out the size of A.
 - ▶ Find the number of elements of A.
 - ▶ Find out the number of elements of A in the “longest” dimension.
 - ▶ Find out the number of dimensions of A.



Squeeze

- Function `squeeze` removes dimension of an array with length 1.
- If the input is scalar, vector or array without any dimension of the length 1, the output is identical to the input.





Function gallery

- ▶ Function enabling to create a vast set of matrices that can be used for MATLAB code testing.
- ▶ Most of the matrices are special-purpose.
 - ▶ Function `gallery` offers significant coding time reduction for advanced MATLAB users.
- ▶ See: doc `gallery`
- ▶ Try for instance:

```
gallery('pei', 5, 4)  
gallery('leslie', 10)  
gallery('clement', 8)
```

Questions?

A8B17CAS

`miloslav.capek@fel.cvut.cz`

September 26

Winter semester 2025/26

This document has been created as a part of A8B17CAS course.
Apart from educational purposes at CTU in Prague, this document may be reproduced, stored, or transmitted only with the prior permission of the authors.