

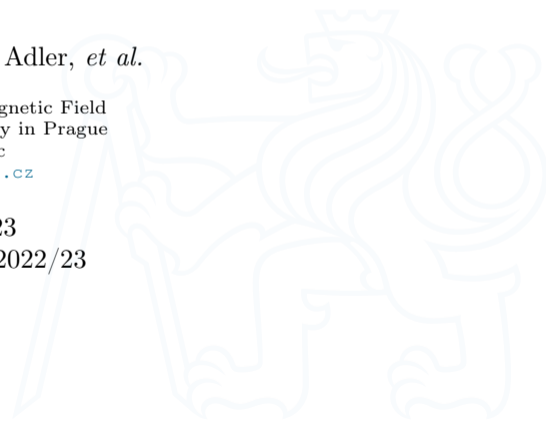
Lecture 3: Indexing, Relational and Logical Operators

B0B17MTB, BE0B17MTB – MATLAB

Miloslav Čapek, Viktor Adler, *et al.*

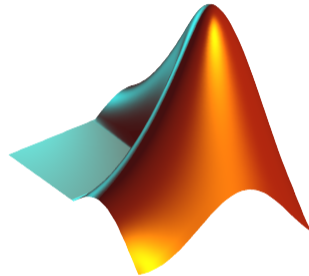
Department of Electromagnetic Field
Czech Technical University in Prague
Czech Republic
matlab@fel.cvut.cz

March 6, 2023
Summer semester 2022/23





1. Indexing
2. Relational Operators
3. Logical Operators
4. Exercises





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 arrays.
- ▶ Two ways of accessing matrices/vectors are distinguished.
 - ▶ Access using round brackets “()”.
 - ▶ Matrix indexing: refers to position of elements in a matrix.
 - ▶ Access using square brackets “[]”.
 - ▶ Matrix concatenation: refers to element’s order in a matrix.



Indexing in MATLAB I.

- ▶ 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 pointer end.

$$\mathbf{N}_1 = \begin{bmatrix} -5 \\ 0 \\ 5 \end{bmatrix}$$

$$\mathbf{N}_2 = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 4 & 6 & 8 & 10 \\ 2 & 3 & 5 & 7 & 11 \end{bmatrix}$$

$$\mathbf{N}_3 = \begin{bmatrix} 11 & 12 & 13 & 14 \\ 22 & 24 & 26 & 28 \\ 33 & 36 & 39 & 42 \\ 44 & 48 & 52 & 56 \end{bmatrix}$$

```
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(3:-1:1)
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)]
```



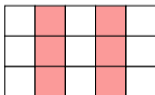
Indexing in MATLAB II.

- ▶ Remember the meaning of end and the application of colon operator “:”.
- ▶ Try to:

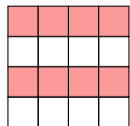
- ▶ Flip the elements of the vector \mathbf{N}_1 without use of `flipplr/flipud` functions.



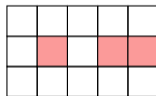
- ▶ Select only the even columns of \mathbf{N}_2 .



- ▶ Select only the odd rows of \mathbf{N}_3 .



- ▶ Select 2nd, 4th and 5th column of 2nd row of \mathbf{N}_2 .



- ▶ Create matrix \mathbf{A} of size 4×3 containing numbers 1 to 12 (row-wise, from left to right).



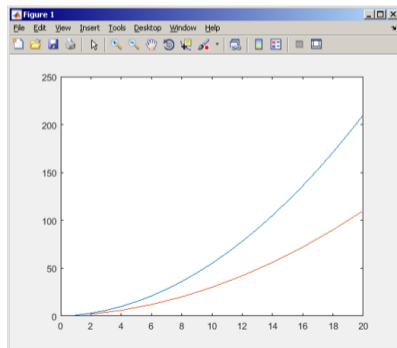
Indexing in MATLAB III.

- ▶ Calculate cumulative sum \mathbf{S} of a vector \mathbf{x} consisting of integers from 1 to 20.
- ▶ Search MATLAB help to find the appropriate function (*cumulative sum*).

$$\mathbf{x} = (1 \quad 2 \quad \dots \quad 20)$$

$$\mathbf{S} = (1 \quad 1+2 \quad \dots \quad 1+2+\dots+20)$$

- ▶ Calculate cumulative sum \mathbf{L} of even element of the vector \mathbf{x} .
- ▶ What is the value of the last element of vector \mathbf{L} ?



Indexing in MATLAB IV.



- Which one of the following returns corner elements of a matrix **A** (10×10)?

```
A([1, 1], [end, end])  
A({[1, 1], [1, end], [end, 1], [end, end]})  
A([1, end], [1, end])  
A(1:end, 1:end)
```



Deleting Elements of a Matrix

- ▶ Empty matrix is a crucial concept in deleting elements of a matrix.

```
T = [];
```

- ▶ We want to:

- ▶ Remove 2nd row of a matrix **A**.

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

- ▶ Remove 3rd column of a matrix **A**.

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

- ▶ Remove 1st, 2nd and 5th column of a matrix **A**.

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




Adding and Replacing Elements of a Matrix

- ▶ We want to replace:

- ▶ 3rd column of a matrix **A** (of size $M \times N$) by a vector **x** (length M).

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

- ▶ 2nd, 4th and 5th row of a matrix **A** by three rows of a matrix **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

- ▶ 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(:, 1, end) = []
A(:, [1:end]) = []
A(:, [1 end]) = []
```

- ▶ Replace 2nd, 3rd and 5th row of matrix **A** by 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:

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

1	0	0	1	1	1	2	0	0
0	1	0	1	1	1	0	3	0
0	0	1	1	1	1	0	0	5

- ▶ Replace elements in the first two rows and columns of the first sheet of the array (*i.e.*, the matrix $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ with NaN elements).



Linear Indexing I.

- ▶ Elements of an array of arbitrary number of dimensions and arbitrary size can be referred using simple index.
 - ▶ Indexing takes place along the main dimension (column-wise) then along the secondary dimension (row-wise) etc.

A = magic(3)

A =

8	1	6
3	5	7
4	9	2

A(1:end)
A(:)

8	1	6
3	5	7
4	9	2

Diagram illustrating column-wise indexing. Red dashed arrows point from the top of each column to the bottom. Green solid arrows point from the bottom of each column to the bottom of the column. A blue arrow points from the matrix to the linear indexing result.

8
3
4
1
5
9
6
7
2

A([1 5])

8	1	6
3	5	7
4	9	2

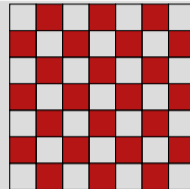
A([1 5], :)

Index in position 1
exceeds array bounds
(must not exceed 3).

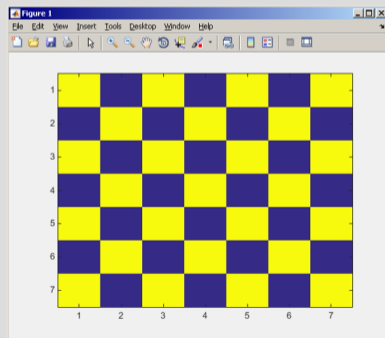


Linear Indexing II.

- ▶ Consider following matrix: $M = \text{ones}(7)$.
 - ▶ We set all the red-highlighted elements to zero:



```
M(2:2:end) = 0;
imagesc(M);
```





Matrix Indexing Using Own Values

- ▶ Create matrix **A**

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

- ▶ First think about what will be the result of the following operation and only then carry it out

```
B = A(A);
```

- ▶ Does the result correspond to what you expected?
 - ▶ Can you explain why the result looks the way it looks?
 - ▶ Notice the interesting mathematical properties of the matrices **A** and **B**.
 - ▶ Are you able to estimate the evolution? $C = B(B)$
- ▶ Try similar process for $N = 3$ or $N = 5$.



Linear Indexing III. - ind2sub, sub2ind

- ▶ `ind2sub` recalculates linear index to subscript corresponding to size and dimensions 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]
```

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

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



Linear Indexing IV.

- ▶ 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, 8]`,
 - ▶ and therefore
 - ▶ `ind = [2, 8, 5, 14]`.

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




Linear Indexing V.

- ▶ Consider following matrix:

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

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



Relational Operators I.

- ▶ To find out, to compare, **whether “something” is greater than, less than, equal to, etc.**
- ▶ The result of the comparison is always either
 - ▶ positive (`true`), logical one “1”,
 - ▶ negative (`false`), logical zero “0”.
- ▶ All relation operators are vector-wise.
 - ▶ It is also possible to compare vector vs. vector, matrix vs. matrix, ...
- ▶ Often in combination with logical operators (*see later*)
 - ▶ Multiple relational operators can be applied to complex expressions.

<code>></code>	greater than
<code>>=</code>	greater than or equal to
<code><</code>	less than
<code><=</code>	less than or equal to
<code>==</code>	equal to
<code>~=</code>	not equal to



Relational Operators II.

- ▶ Having the vector $\mathbf{G} = \left(\frac{\pi}{2} \quad \pi \quad \frac{3\pi}{2} \quad 2\pi \right)$, find elements of \mathbf{G} that are
 - ▶ greater than π ,
 - ▶ less than or equal to π ,
 - ▶ not equal to π .
 - ▶ Try similar operations for $\mathbf{H} = \mathbf{G}^T$.
- ▶ Try to use relational operators in case of matrices and scalars as well.
 - ▶ Find out whether $\mathbf{V} \geq \mathbf{U}$:
 - ▶ $\mathbf{V} = \begin{pmatrix} -\pi & \pi & 1 & 0 \end{pmatrix}$,
 - ▶ $\mathbf{U} = \begin{pmatrix} 1 & 1 & 1 & 1 \end{pmatrix}$.



Relational Operators III.

- ▶ Find out the results of following relations.
 - ▶ Try to interpret the results.

```
2 < 1 ~= 1 % ???
```

```
r = 1/2;  
0 < r < 1 % ???
```

```
(1 > A) <= true
```



Logical Operators I.

- ▶ To find out, **whether particular condition is fulfilled.**
- ▶ The result is always either
 - ▶ positive (`true`), logical one “1”,
 - ▶ negative (`false`), logical zero “0”.
- ▶ `all`, `any` is used to convert logical array into a scalar.
- ▶ MATLAB interprets any numerical value except 0 as `true`.
- ▶ All logical operators are vector-wise.
 - ▶ It is also possible to compare vector vs. vector, matrix vs. matrix, ...
- ▶ Function `is*` extends possibilities of logical expressions.
 - ▶ We will see later

<code>&</code>	and
<code> </code>	or
<code>~</code>	not
	xor
	all
	any



Logical Operators II.

- ▶ Assume a vector of 10 random numbers ranging from -10 to 10.

```
a = 20*rand(10, 1) - 10
```

- ▶ Following command returns `true` for elements fulfilling the condition.

```
a < -5 % relation operator
```

- ▶ Following command returns values of those elements fulfilling the condition (logical indexing).

```
a(a < -5)
```

- ▶ Following command puts value of -5 to the position of elements fulfilling the condition.

```
a(a < -5) = -5
```

- ▶ Following command sets value of the elements in the range from -5 to 5 equal to zero (opposite to thresholding).

```
a(a > -5 & a < 5) = 0
```

- ▶ Thresholding function (values below -5 set equal to -5, values above 5 set equal to 5).

```
a(a < -5 | a > 5) = ...  
sign(a(a < -5 | a > 5))*5
```



Logical Operators III.

- ▶ Determine which of the elements of the vector $\mathbf{A} = \left[\frac{\pi}{2} \quad \pi \quad \frac{3\pi}{2} \quad 2\pi \right]$ fulfill following condition.
 - ▶ Which elements are equal to π or are equal to 2π .
 - ▶ Pay attention to the type of the results (=logical values true/false).
 - ▶ Which elements are greater than $\frac{\pi}{2}$ and at the same time are not equal to 2π .
- ▶ Group elements from the previous condition with vector \mathbf{A} .



Logical Operators IV.

- ▶ Create a row vector in the interval from 1 to 20 with step of 3.
 - ▶ Create the vector filled with elements from the previous vector that are:
 - ▶ greater than 10
 - ▶ and at the same time
 - ▶ less than 16.
 - ▶ Use logical operators.



Logical Operators V.

- ▶ Create matrix **M** (`M = magic(3)`) and answer following questions using functions `all` and `any`.

- ▶ In which of the columns all elements are greater than 2?

$$\text{any}\left(\begin{bmatrix} 0 & 1 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix}\right) = [1 \quad 1 \quad 1],$$

- ▶ In which of the rows there is at least one element greater than or equal to 8?

$$\text{all}\left(\begin{bmatrix} 0 & 1 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix}\right) = [0 \quad 1 \quad 0],$$

- ▶ Does the matrix **M** contain only positive numbers?

$$\text{any}(\text{all}\left(\begin{bmatrix} 0 & 1 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix}\right)) = \text{any}([0 \quad 1 \quad 0]) = 1$$



Logical Operators VI.

- ▶ In the case we need to compare scalar values only then “short-circuited” evaluation can be used.
- ▶ Evaluation keeps on going until the point where it makes no sense to continue
 - ▶ *e.g.*, when evaluating

```
clear;  
a = true;  
b = false;  
a && b && c && d
```

- ▶ However:

```
clear;  
a = true;  
b = true;  
a && b && c && d
```

- ▶ There are no problems with undefined variables `c` and `d`, because the execution is terminated before evaluating those variables.
- ▶ This is terminated with error ...
`Unrecognized function or variable 'c'.`



Logical Operators VII.

- ▶ Find out the result of the following operation and interpret it.

```
~(~[1 2 0 -2 0])
```

- ▶ Test whether variable b is not equal to zero and then test whether at the same time $a/b > 3$.
 - ▶ Following operation tests whether both conditions are fulfilled while avoiding division by zero.

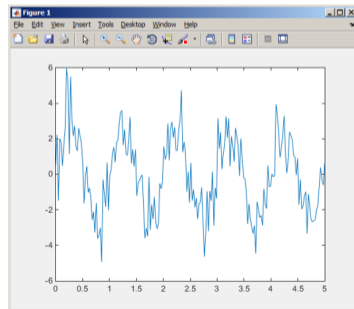
Exercises



Exercise I.

- ▶ Consider signal: $s(t) = \sqrt{2\pi} \sin(2\omega_0 t) + n(\mu, \sigma)$, $\omega_0 = \pi$, where the mean and standard deviation of normal distribution n are: $\mu = 0$ (mu = 0), $\sigma = 1$ (sigma = 1).
 - ▶ Create time dependence of the signal spanning over $N = 5$ periods of the signal using $V = 40$ samples per period.
 - ▶ One period is $T = 1$ such that $t \in [kT, (k + N)T]$, $k \in \mathbb{Z}^0$ (choose k equal for instance to 0).
 - ▶ The function $n(\mu, \sigma)$ has the following MATLAB syntax:

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



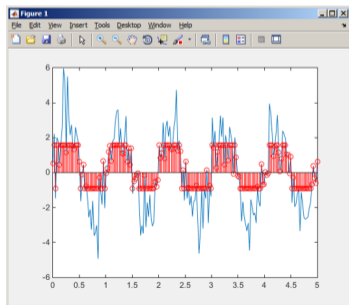


Exercise II.

- ▶ Apply threshold function to generated signal from the previous exercise to limit its maximum and minimum value:
 - ▶ The result is vector `sp_t`.
 - ▶ Use function `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');
```

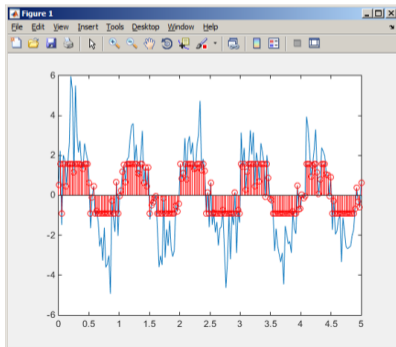
$$s_p(t) = \begin{cases} s_{\min} \Leftrightarrow s(t) < s_{\min} \\ s_{\max} \Leftrightarrow s(t) > s_{\max} \\ s(t) \dots \text{otherwise} \end{cases} \quad \begin{aligned} s_{\min} &= -\frac{9}{10} \\ s_{\max} &= \frac{\pi}{2} \end{aligned}$$





Exercise III.

- ▶ Recall the signal from Exercise I.
 - ▶ Try again to limit the signal by values s_{\min} and s_{\max} .
 - ▶ Use relational operators ($>/<$) and logical indexing ($s(a>b) = c$) instead of functions `min` and `max`.
 - ▶ Solve the task item-by-item.



```
N = 5; V = 40;
t = linspace(0, N, N*V);
s_t = randn(1, N*V) +
sqrt(2*pi)*sin(2*pi*t);
```



Exercise IV.a

- ▶ Create a script to calculate compound interest¹.
 - ▶ 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 n , r or k .
- ▶ Check your results (for $A = 1000$, $n = 12$, $k = 15$, $r = 0.1$ is $P = 10.7461$).

¹Interest from the prior period is added to principal.

Exercise IV.b



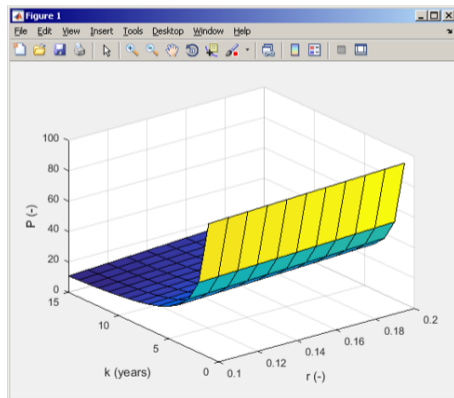
- ▶ Try to vectorize the code, both for r and k .
- ▶ Use scripts for future work with MATLAB.
 - ▶ Bear in mind, however, that parts of the code can be debugged using command line.

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



Exercise IV.c

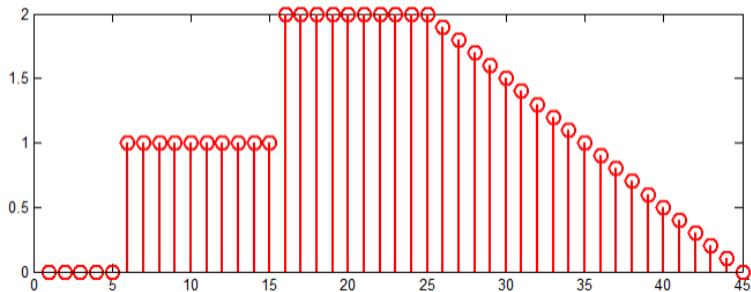
- ▶ Vectorized code for both r and k .
 - ▶ The compatible size array feature used.
 - ▶ `surf` created 3D surface plot.





Exercise V.a

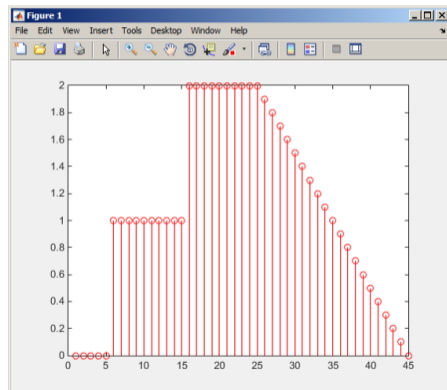
- ▶ Generate vector containing following sequence.
 - ▶ Note the x -axis (interval, number of samples).
 - ▶ Split the problem into several parts to be solved separately.
 - ▶ Several ways how to solve the problem.
 - ▶ Use `stem(x)` instead of `plot(x)` for plotting.
- ▶ Try to generate the same signal beginning with zero ...





Exercise V.b

- ▶ Generate vector containing following sequence.





Exercise VI.

- ▶ Consider following matrix $\mathbf{A} = \begin{bmatrix} 1 & 1 & 2 \\ 2 & 3 & 5 \end{bmatrix}$.
- ▶ Write an expression testing whether all elements of \mathbf{A} are positive and at the same time all elements of the first row are integers.



Exercise VII.a

- ▶ Reflection coefficient S_{11} of a one-port device of impedance Z is given by:

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

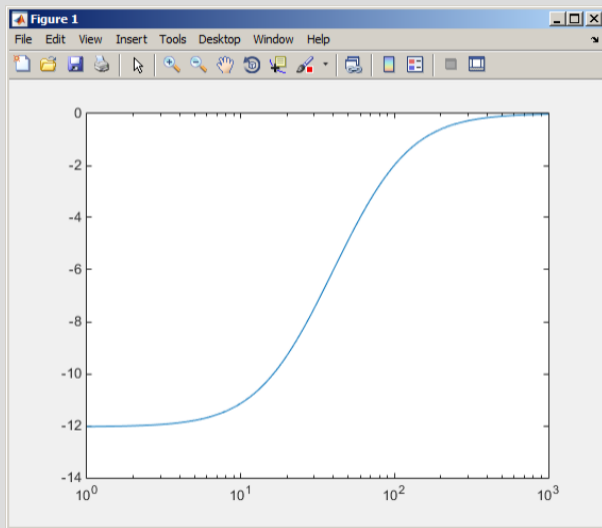
where $Z_0 = 50\Omega$ and $Z = R + jX$.

- ▶ Calculate and depict the dependence of S_{11} for $R = 30\Omega$ and X on the interval $[1, 1000]$ with 100 evenly spaced points in logarithmic scale.
- ▶ Use the code below and correct errors in the code. Correct solution generates plot depicted on the next slide.

```
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 VII.b



Questions?

B0B17MTB, BE0B17MTB – MATLAB
matlab@fel.cvut.cz

March 6, 2023
Summer semester 2022/23

This document has been created as a part of B(E)0B17MTB 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.

Acknowledgement: Filip Kozák, Pavel Valtr, Michal Mašek, and Vít Losenický.