

# Lecture 0: Introduction and Terminology

A8B17CAS

Miloslav Čapek

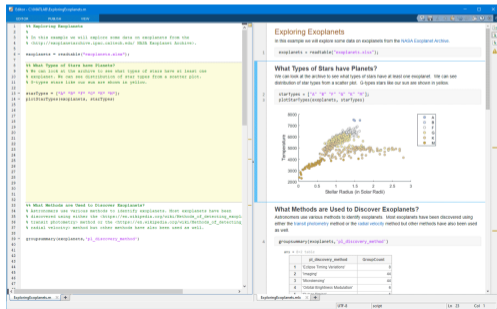
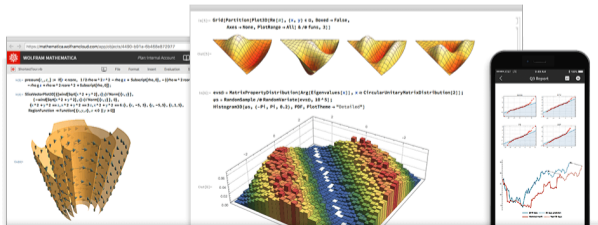
Department of Electromagnetic Field  
Czech Technical University in Prague  
Czech Republic  
[miloslav.capek@fel.cvut.cz](mailto:miloslav.capek@fel.cvut.cz)

September 23  
Winter semester 2024/25





1. Programming and Numerical Computing Platforms
2. Why to Learn MATLAB & MATHEMATICA?
3. Basic Terms and Terminology
4. Example: A System of Linear Equations



## Programming and Numerical Computing Platforms...

*...help us with numerical and data analysis, i.e., to find an exact or approximate solution to numerical problems, analyze, modify, and present data, and develop algorithms and codes.*

### Common characteristics

- ▶ high-level programming languages (*4th generation*),
- ▶ excellent for “fast prototyping”.



# What Platforms Do We Have?

- ▶ They are many.
- ▶ From free & open-source to large & expensive systems.

▶ [Software classification \(wiki\)](#)

▶ [List of numerical-analysis software \(wiki\)](#)



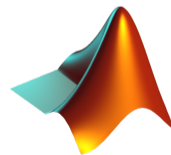
# What Platforms Do We Have?

- ▶ They are many.
- ▶ From free & open-source to large & expensive systems.

▶ [Software classification \(wiki\)](#)

▶ [List of numerical-analysis software \(wiki\)](#)

- ▶ We will focus on MATLAB and MATHEMATICA.



## Why to learn MATLAB & MATHEMATICA?

- ▶ They are worldwide standards.
- ▶ They are used by thousands of universities worldwide.
- ▶ License used by a plethora of corporations in aviation, biotechnology, electronics, cybernetics, mechanical engineering, finance, ...

## Why to learn MATLAB & MATHEMATICA?

- ▶ They are worldwide standards.
- ▶ They are used by thousands of universities worldwide.
- ▶ License used by a plethora of corporations in aviation, biotechnology, electronics, cybernetics, mechanical engineering, finance, ...

## Where we can use them?

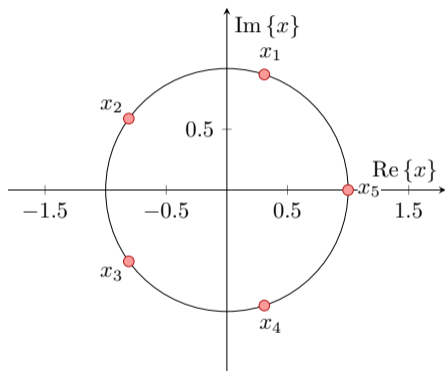
- ▶ Data processing and visualization during laboratory exercises.
- ▶ When elaborating diploma works.
- ▶ Seminar exercises (signals, algorithm development, ...).
- ▶ Theory verification (mathematics and physics classes, electromagnetic field, electronic circuits, ...).
- ▶ Studying aboard (Erasmus, Sokrates).
- ▶ In daily professional live.
- ▶ “**everywhere**” :)



# Polynomial Roots (Problem Assignment)

Find polynomial roots  $x_n$  of

$$x^5 = 1.$$



Roots  $x_n$  of polynomial  $x^5 = 1$  depicted in the complex plane.





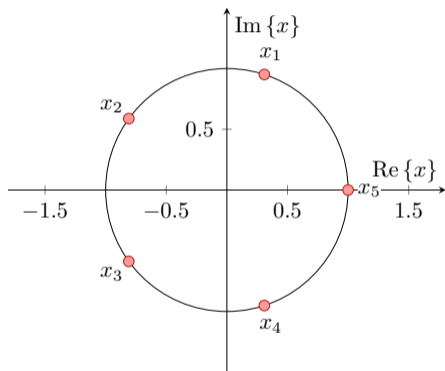
# Polynomial Roots (Problem Assignment)

Find polynomial roots  $x_n$  of

$$x^5 = 1.$$

- We are lucky! All roots lie on the unitary circle in a complex plane, *i.e.*,

$$x_n = \exp \left\{ j \frac{2\pi n}{5} \right\} = (-1)^{(n-1)/5}, \quad i \in \{1, \dots, 5\}.$$



Roots  $x_n$  of polynomial  $x^5 = 1$  depicted in the complex plane.



# Polynomial Roots (Problem Assignment)

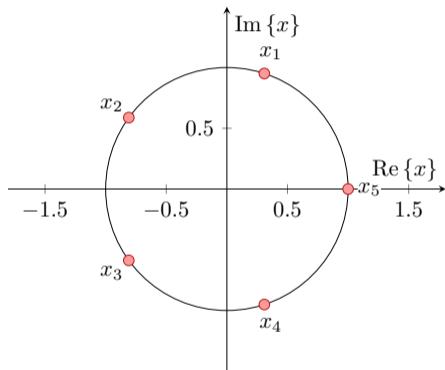
Find polynomial roots  $x_n$  of

$$x^5 = 1.$$

- ▶ We are lucky! All roots lie on the unitary circle in a complex plane, *i.e.*,

$$x_n = \exp \left\{ j \frac{2\pi n}{5} \right\} = (-1)^{(n-1)/5}, \quad i \in \{1, \dots, 5\}.$$

- ▶ However, in general, quintic and higher polynomials are unsolvable via radicals (Abel–Ruffini theorem). Try to solve  $x^5 - x - 1 = 0 \dots$  impossible!



Roots  $x_n$  of polynomial  $x^5 = 1$  depicted in the complex plane.



## Polynomial Roots (Problem Assignment)

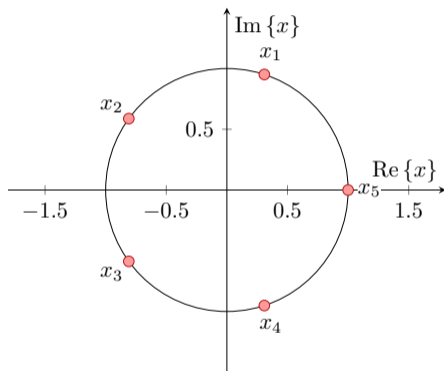
Find polynomial roots  $x_n$  of

$$x^5 = 1.$$

- ▶ We are lucky! All roots lie on the unitary circle in a complex plane, *i.e.*,

$$x_n = \exp \left\{ j \frac{2\pi n}{5} \right\} = (-1)^{(n-1)/5}, \quad i \in \{1, \dots, 5\}.$$

- ▶ However, in general, quintic and higher polynomials are unsolvable via radicals (Abel–Ruffini theorem). Try to solve  $x^5 - x - 1 = 0 \dots$  impossible!
- ▶ Consequently, a numerical solution is required!



Roots  $x_n$  of polynomial  $x^5 = 1$  depicted in the complex plane.



# Polynomial Roots (Problem Assignment)

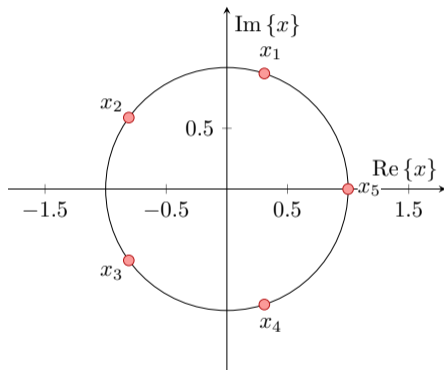
Find polynomial roots  $x_n$  of

$$x^5 = 1.$$

- ▶ We are lucky! All roots lie on the unitary circle in a complex plane, *i.e.*,

$$x_n = \exp\left\{j\frac{2\pi n}{5}\right\} = (-1)^{(n-1)/5}, \quad i \in \{1, \dots, 5\}.$$

- ▶ However, in general, quintic and higher polynomials are unsolvable via radicals (Abel–Ruffini theorem). Try to solve  $x^5 - x - 1 = 0 \dots$  impossible!
- ▶ Consequently, a numerical solution is required!
- ▶ Workflow: set up the problem, visualize if needed, solve, check, and present the data (results).



Roots  $x_n$  of polynomial  $x^5 = 1$  depicted in the complex plane.



# Polynomial Roots (Problem Solution: MATLAB)

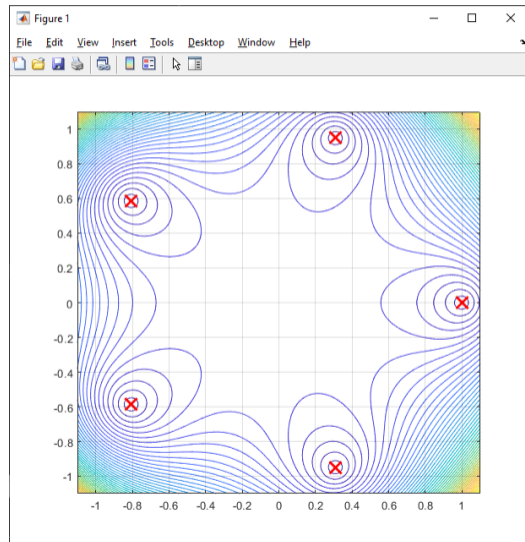
```
clear;

% Set up polynomial and find roots
p = [1 0 0 0 0 -1];
r = roots(p);

% Calculate data for visualisation
x = -1.1:1/100:1.1;
[Re, Im] = meshgrid(x, x. ');
X = complex(Re, Im);
F = polyval(p, X);

% Plot the complex plane and the roots
contour(x, x, abs(F), 51);
hold on;
grid on;
plot(real(r), imag(r), 'rx', ...
      'MarkerSize', 15, 'LineWidth', 2);

% Analytical check
exp(1j*(0:(2*pi/5):(4*2*pi/5)))
```





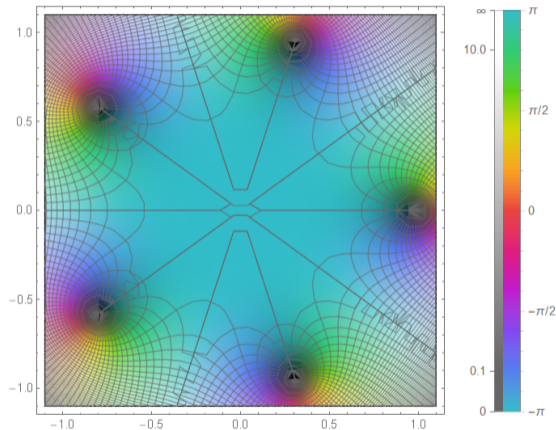
# Polynomial Roots (Problem Solution: MATHEMATICA)

```
In[1]= p = x^5 - 1;
```

```
  r = Roots[p == 0, x]
```

```
Out[2]= x == 1 || x == (-1)^(2/5) || x == (-1)^(4/5) || x == -(-1)^(1/5) || x == -(-1)^(3/5)
```

```
In[3]= ComplexPlot[p, {x, -1.1 - 1.1 I, 1.1 + 1.1 I}, Mesh -> 51,  
  PlotLegends -> Automatic, ColorFunction -> "GlobalAbs"]
```





# Analytical (Symbolic) × Numerical Evaluation

- ▶ “Analytical” solution to a problem is exact and obtained by methods of symbolic manipulation, derived using analysis.
  - ▶ When you do an “analytical solution” you answer to a whole set of problems, *e.g.*,  $2a/a = 2 \quad \forall a$ .
  - ▶ It provides generally valid results and often reveals the properties of the problem.



# Analytical (Symbolic) × Numerical Evaluation

- ▶ “Analytical” solution to a problem is exact and obtained by methods of symbolic manipulation, derived using analysis.
  - ▶ When you do an “analytical solution” you answer to a whole set of problems, *e.g.*,  $2a/a = 2 \quad \forall a$ .
  - ▶ It provides generally valid results and often reveals the properties of the problem.
- ▶ “Numerical” solution to a problem usually indicates an approximate solution obtained by methods of numerical analysis.
  - ▶ When you do a “numerical solution” you are generally only getting one answer, *e.g.*,  $4/2 = 2$ .
  - ▶ A particular solution that can be achieved for a large set of problems than the “analytical solution”.





# Analytical (Symbolic) $\times$ Numerical Evaluation

- ▶ “Analytical” solution to a problem is exact and obtained by methods of symbolic manipulation, derived using analysis.
  - ▶ When you do an “analytical solution” you answer to a whole set of problems, *e.g.*,  $2a/a = 2 \quad \forall a$ .
  - ▶ It provides generally valid results and often reveals the properties of the problem.
- ▶ “Numerical” solution to a problem usually indicates an approximate solution obtained by methods of numerical analysis.
  - ▶ When you do a “numerical solution” you are generally only getting one answer, *e.g.*,  $4/2 = 2$ .
  - ▶ A particular solution that can be achieved for a large set of problems than the “analytical solution”.

**Example:** The following doublets of MATHEMATICA commands lead to different results:  
 $2/4 = 1/2$  vs.  $N[2/4] = 0.5$ , and  $Pi = \pi$  vs.  $N[Pi, 5] = 3.1416$ .

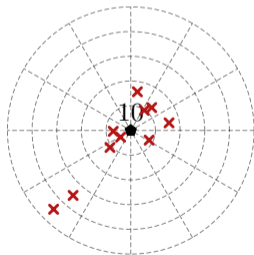


# A Note on Difference Between Accuracy and Precision

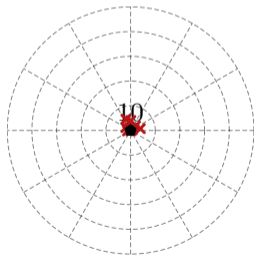
**Be accurate:** You are as close to the chosen goal/result as possible.

**Be precise:** You can repeat the evaluation/experiment as similarly as possible.

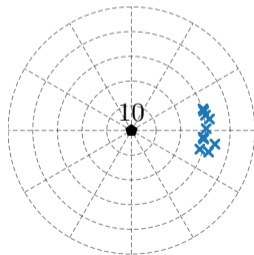
**Example:** Archery



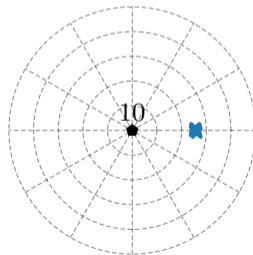
low accuracy



high accuracy



low precision



high precision



# Compiled × Interpreted Language

## Compiled language

- ▶ After compilation, the code is expressed in the instructions of the target machine.
- ▶ Compilation takes some time, but the final code is generally faster than the interpreted code (with a good compiler).



# Compiled × Interpreted Language

## Compiled language

- ▶ After compilation, the code is expressed in the instructions of the target machine.
- ▶ Compilation takes some time, but the final code is generally faster than the interpreted code (with a good compiler).

## Interpreted language

- ▶ The code is executed line by line.
- ▶ Easier coding as compared to a compiled language.
- ▶ Allows execution of separate lines in arbitrary order (with easy on-fly modifications).
- ▶ All the debugging (and errors) occurs at run-time.



# Compiled × Interpreted Language

## Compiled language

- ▶ After compilation, the code is expressed in the instructions of the target machine.
- ▶ Compilation takes some time, but the final code is generally faster than the interpreted code (with a good compiler).

## Interpreted language

- ▶ The code is executed line by line.
- ▶ Easier coding as compared to a compiled language.
- ▶ Allows execution of separate lines in arbitrary order (with easy on-fly modifications).
- ▶ All the debugging (and errors) occurs at run-time.

There are advanced techniques sharing good properties of compiled and interpreted code.

- ▶ *e.g.*, JIT in MATLAB



# Type of Errors

In general, there are many types of errors:

**Truncation** finite numerical precision

**Discretization** continuous problem is solved point-wise, derivatives are replaced with differences, etc.

**Modeling** intentional simplification of the model

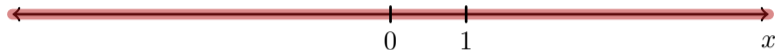
**Empirical constant** physical constants are used with a certain precision

**Input** typos, intentionally incorrect input data, inaccurate measurement



# Floating Point Numbers I.

Continuous axis or real numbers:



- ▶ Infinite resolution cannot be stored in computers based on finite arithmetic.



# Floating Point Numbers I.

Continuous axis or real numbers:



- ▶ Infinite resolution cannot be stored in computers based on finite arithmetic.
- ▶ IEEE 754 (Standard for Floating-Point Arithmetic)
  - ▶ single precision (32 bits = 1 bit sign + 8 bits exponent + 23 bit mantissa)
  - ▶ double precision (64 bits = 1 bit sign + 11 bits exponent + 52 bit mantissa)

Floating-point axis (in the vicinity of the number 1):







## Floating Point Numbers II.

Decimal vs. sign-exponent-mantissa formats:

▶  $+2^0 = 1$

= 0 0111111111 000

▶  $+2^0 \times (1 + 2^{-52}) \approx 1.0000000000000002$

= 0 0111111111 001





## Floating Point Numbers II.

Decimal vs. sign-exponent-mantissa formats:

►  $+2^0 = 1$

= 0 0111111111 000

►  $+2^0 \times (1 + 2^{-52}) \approx 1.0000000000000002$

= 0 0111111111 001

Be aware of finite-precision arithmetic:

```
>> cos(pi/2) = 6.123233995736766e-17  
>> eps(1) = 2.220446049250313e-16  
>> log2(eps(1)) = -52
```

See more at [► Floating Point Numbers \(C. Moler\)](#) and [► Numerics and Error Analysis \(Stanford\)](#).



# Nomenclature

Let us agree on some basic notation.

- ▶ Scalars  $a$ , vectors  $\mathbf{a}$ , matrices  $\mathbf{A}$ , arrays,
- ▶ equations  $\times$  expressions,
- ▶ algebraic, logical, relational, bit-wise, set (and other) operators.

Distinguish between mathematical notation and syntax in a given software!

- ▶ For example, a vector  $\mathbf{v} = [1 \ 2 \ 3]^T$  is created in MATLAB as `v = [1; 2; 3]`.
- ▶ Notice that all PDF materials are made in L<sup>A</sup>T<sub>E</sub>X (Beamer) in Overleaf.
- ▶ Graphics (whenever possible) is made in L<sup>A</sup>T<sub>E</sub>X package TikZ.



# System of Linear Equations

Consider a generic system of linear equations of the form:

$$\begin{aligned}a_{11}x_1 + a_{12}x_2 &= b_1, \\ a_{21}x_1 + a_{22}x_2 &= b_2,\end{aligned}\tag{1}$$

which can be written in algebraic form as

$$\mathbf{Ax} = \mathbf{b},\tag{2}$$

where

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}.$$



# System of Linear Equations

Consider a generic system of linear equations of the form:

$$\begin{aligned}a_{11}x_1 + a_{12}x_2 &= b_1, \\ a_{21}x_1 + a_{22}x_2 &= b_2,\end{aligned}\tag{1}$$

which can be written in algebraic form as

$$\mathbf{Ax} = \mathbf{b},\tag{2}$$

where

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}.$$

Considering the matrix  $\mathbf{A}$  is non-singular, the solution is found as

$$\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}.\tag{3}$$



# System of Linear Equations – An Interpretation

Let us be more specific:

$$\begin{bmatrix} 1 & -1 \\ 0.5 & 0.5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

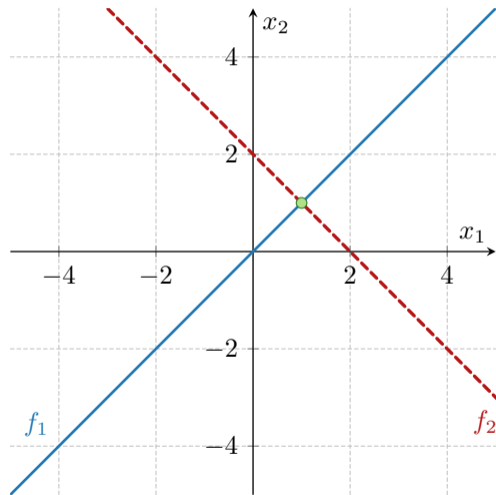
Visual depiction by two lines, *i.e.*,

$$f_1 : x_1 = x_2,$$

$$f_2 : x_1 = 2 - x_2.$$

MATLAB:

```
>> A = [1 -1; 1/2 1/2]
>> b = [0; 1]
>> x = A\b
```



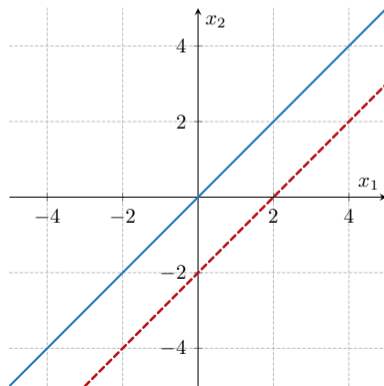
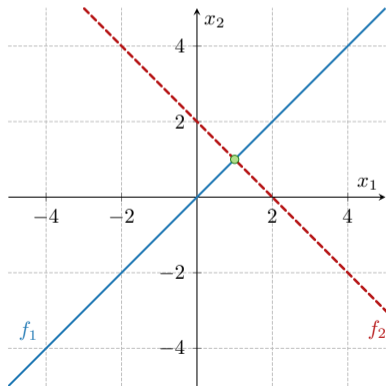
## System of Linear Equations – Well- and Ill-Conditioned Problems



Looks like a piece of cake, however, be always careful...

$$\begin{bmatrix} 1 & -1 \\ 0.5 & 0.5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \Rightarrow \mathbf{x} = \begin{bmatrix} 1 \\ 1 \end{bmatrix},$$

$$\begin{bmatrix} 1 & -1 \\ 0.5 & -0.5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \Rightarrow \mathbf{x} = \begin{bmatrix} ? \\ ? \end{bmatrix}.$$





# System of Linear Equations – An Analysis of the Problem



- ▶ We will need: determinant,  $\det(\mathbf{A})$ , matrix inversion,  $\mathbf{A}^{-1}$ , dot product,  $\mathbf{u} \cdot \mathbf{v}$ , second norm of a vector,  $\|\mathbf{u}\|$ , condition number,  $\text{cond}(\mathbf{A})$ .
- ▶ We have to investigate: how to approach the problem, the existence of a solution, accuracy of the results.
- ▶ We can try to find: a symbolic solution, extend the problem, and solve special cases.



# System of Linear Equations – An Analysis of the Problem

- ▶ We will need: determinant,  $\det(\mathbf{A})$ , matrix inversion,  $\mathbf{A}^{-1}$ , dot product,  $\mathbf{u} \cdot \mathbf{v}$ , second norm of a vector,  $\|\mathbf{u}\|$ , condition number,  $\text{cond}(\mathbf{A})$ .
- ▶ We have to investigate: how to approach the problem, the existence of a solution, accuracy of the results.
- ▶ We can try to find: a symbolic solution, extend the problem, and solve special cases.

$$\det(\mathbf{A}) = a_{11}a_{22} - a_{12}a_{21} \quad \text{>> det(A)} \quad \mathbf{A}^{-1} = \frac{1}{\det(\mathbf{A})} \begin{bmatrix} +a_{22} & -a_{12} \\ -a_{21} & +a_{11} \end{bmatrix} \quad \text{>> inv(A)}$$

$$\mathbf{u} \cdot \mathbf{v} = \sum_i u_i v_i \quad \text{>> dot(u,v)} \quad \|\mathbf{u}\| = \sqrt{\mathbf{u} \cdot \mathbf{u}} \quad \text{>> norm(u, 2)}$$

$$\text{cond}(\mathbf{A}) = \|\mathbf{A}\| \|\mathbf{A}^{-1}\| \quad \text{>> cond(A)}$$

# System of Linear Equations – Generalization & Symbolic Solution



- ▶ Whenever possible, try to find a symbolic solution (good for verification and understanding of the problem's properties).

## System of Linear Equations – Generalization &amp; Symbolic Solution



- ▶ Whenever possible, try to find a symbolic solution (good for verification and understanding of the problem's properties).

Let us consider the following generalization  $\mathbf{A} \rightarrow \mathbf{A}(\kappa)$

$$\begin{bmatrix} 1 & -1 \\ 0.5 & 0.5 - \kappa \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad \kappa \in [0, 1],$$

covering all cases from well-defined ( $\kappa = 0$ ) to ill-defined ( $\kappa = 1$ ).

## System of Linear Equations – Generalization &amp; Symbolic Solution



- Whenever possible, try to find a symbolic solution (good for verification and understanding of the problem's properties).

Let us consider the following generalization  $\mathbf{A} \rightarrow \mathbf{A}(\kappa)$

$$\begin{bmatrix} 1 & -1 \\ 0.5 & 0.5 - \kappa \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad \kappa \in [0, 1],$$

covering all cases from well-defined ( $\kappa = 0$ ) to ill-defined ( $\kappa = 1$ ).

$$\det(\mathbf{A}) = 1 - \kappa, \quad \mathbf{x} = \mathbf{A}^{-1}(\kappa) \mathbf{b} = \frac{1}{1 - \kappa} \begin{bmatrix} 0.5 - \kappa & 1 \\ -0.5 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \frac{1}{1 - \kappa} \begin{bmatrix} 1 \\ 1 \end{bmatrix} \Leftrightarrow \kappa \neq 1.$$

System of Linear Equations – A Note on Angle Between  $f_1$  and  $f_2$ 

- ▶ To study the finite-precision performance of MATLAB depending on  $\kappa$ , let us express angle  $\alpha$  between  $f_1$  and  $f_2(\kappa)$ .
- ▶ Two possibilities shown:

System of Linear Equations – A Note on Angle Between  $f_1$  and  $f_2$ 

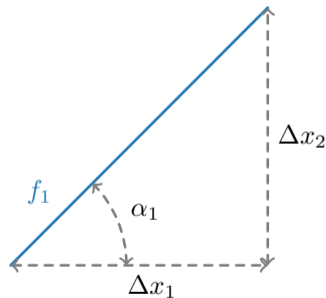
- ▶ To study the finite-precision performance of MATLAB depending on  $\kappa$ , let us express angle  $\alpha$  between  $f_1$  and  $f_2(\kappa)$ .
- ▶ Two possibilities shown:

## I. Trigonometry and a slope of the line:

$$f_1 : x_2 = \frac{b_1 - a_{11}x_1}{a_{12}} \quad \text{and analogously for } f_2(\kappa)$$

$$f_1 : \frac{dx_2}{dx_1} = -\frac{a_{11}}{a_{12}} = \tan(\alpha_1) \quad \text{and analogously for } f_2(\kappa)$$

$$|\alpha| = |\alpha_1 - \alpha_2| = \left| \arctan\left(-\frac{a_{11}}{a_{12}}\right) - \arctan\left(-\frac{a_{21}}{a_{22}}\right) \right|$$



System of Linear Equations – A Note on Angle Between  $f_1$  and  $f_2$ 

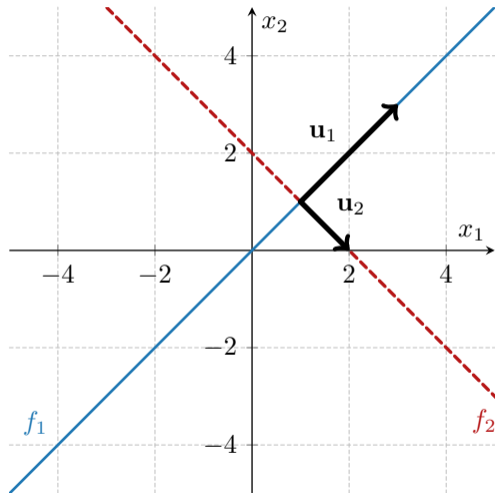
## II. Property of the inner product:

$$\mathbf{u}_1 \cdot \mathbf{u}_2 = \|\mathbf{u}_1\| \|\mathbf{u}_2\| \cos(\alpha)$$

$$|\alpha| = \left| \arccos \left( \frac{\mathbf{u}_1 \cdot \mathbf{u}_2}{\|\mathbf{u}_1\| \|\mathbf{u}_2\|} \right) \right|$$

Normal vectors  $\mathbf{u}_1$  and  $\mathbf{u}_2$  along  $f_1$  and  $f_2$  are

$$\mathbf{u}_1 = c_1 \begin{bmatrix} a_{11} \\ -a_{12} \end{bmatrix}, \quad \mathbf{u}_2 = c_2 \begin{bmatrix} a_{21} \\ -a_{22} \end{bmatrix}.$$





# System of Linear Equations – Parametric Study



- ▶ Since we know the analytical solution to the problem, we can compare MATLAB solution with it (depending on  $\kappa$ ).
- ▶ As we should expect, the accuracy will be dependent on floating-point resolution.

MATLAB (**double** precision):

$$\kappa = 0.999999999000000$$

$$x_1(\kappa) = 1000000028.281932$$

$$x_2(\kappa) = 1000000028.281932$$

$$|\det(\mathbf{A}(\kappa))| = 0.000000001000000$$

$$\text{cond}(\mathbf{A}) = 2500000205.439272$$

MATLAB (**single** precision):

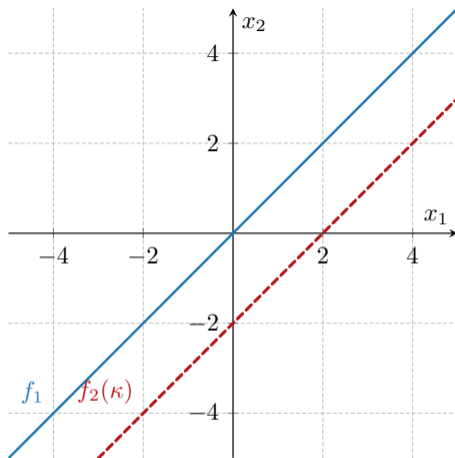
$$\kappa = 0.999999999000000$$

$$x_1(\kappa) = \text{Inf}$$

$$x_2(\kappa) = \text{Inf}$$

$$|\det(\mathbf{A}(\kappa))| = 0.000000000000000$$

$$\text{cond}(\mathbf{A}) = 163117008.0000000$$



MATLAB (double precision):

$$\kappa = 0.999999999000000$$

$$x_1(\kappa) = 1000000028.281932$$

$$x_2(\kappa) = 1000000028.281932$$

$$|\det(\mathbf{A}(\kappa))| = 0.000000001000000$$

$$\text{cond}(\mathbf{A}) = 2500000205.439272$$

MATLAB (single precision):

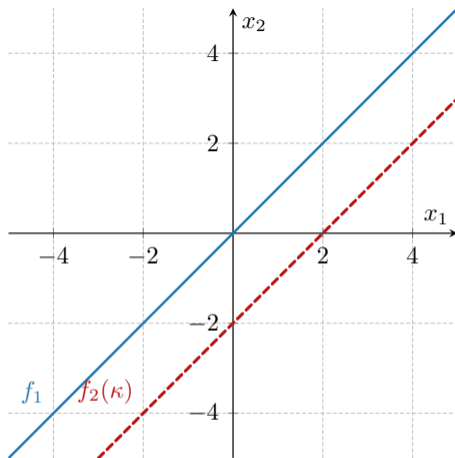
$$\kappa = 0.999999999000000$$

$$x_1(\kappa) = \text{Inf}$$

$$x_2(\kappa) = \text{Inf}$$

$$|\det(\mathbf{A}(\kappa))| = 0.000000000000000$$

$$\text{cond}(\mathbf{A}) = 163117008.0000000$$

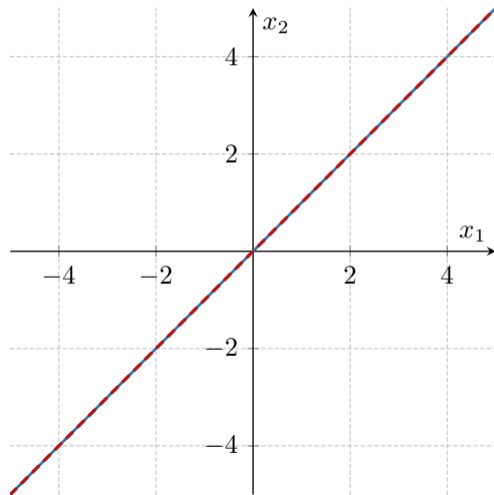


- ▶ Single precision gives finite results up to  $\kappa \approx 1 - 10^{-7}$ , since  $\text{eps}(\text{single}(1)) \approx 1.2 \cdot 10^{-7}$ , *i.e.*,  $\text{abs}(\log_2(\text{eps}(\text{single}(1)))) = 23$ .
- ▶ Double precision gives finite results up to  $\kappa \approx 1 - 10^{-16}$ , since  $\text{eps}(\text{double}(1)) \approx 2.22 \cdot 10^{-16}$ , *i.e.*,  $\text{abs}(\log_2(\text{eps}(\text{double}(1)))) = 52$ .

## System of Linear Equations – On Special Cases



We are yet not finished!





# System of Linear Equations – On Special Cases

- ▶ If the RHS of the problem

$$\begin{bmatrix} 1 & -1 \\ 0.5 & 0.5 - \kappa \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

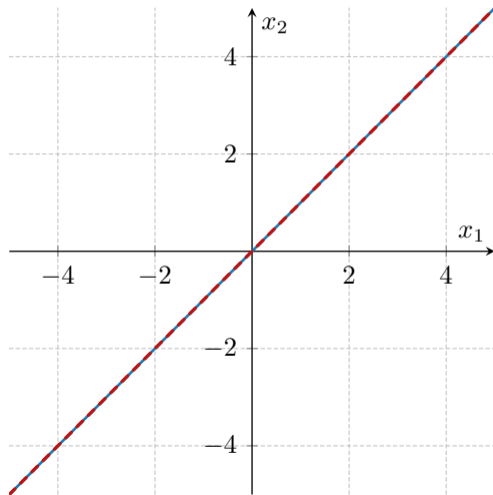
is changed to

$$\begin{bmatrix} 1 & -1 \\ 0.5 & 0.5 - \kappa \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix},$$

we get for  $\kappa \rightarrow 1$

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \frac{1}{1 - \kappa} \begin{bmatrix} 0 \\ 0 \end{bmatrix} \rightarrow \infty \begin{bmatrix} 0 \\ 0 \end{bmatrix} = \begin{bmatrix} \text{NaN} \\ \text{NaN} \end{bmatrix}.$$

- ▶ Remember, sometimes, special cases lead to a very different solution!



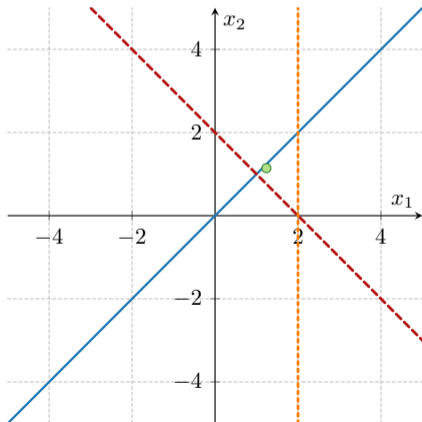
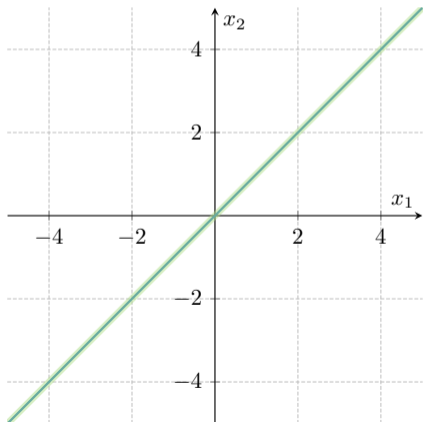
## System of Linear Equations – Under-/Over-determined Problems



$$\begin{bmatrix} 1 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \end{bmatrix}$$

to

$$\begin{bmatrix} 1 & -1 \\ 0.5 & 0.5 \\ 0.5 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}$$



## System of Linear Equations – Under-/Over-determined Problems



How to *a priori* recognize under-/over-determined system?

- ▶ Check rank of the matrix  $\mathbf{A}$ , MATLAB: `rank(A)`.
- ▶ Check the reduced echelon form of the augmented matrix, *i.e.*, `rref([A b])` in MATLAB. Is there any zero row?

Powerful techniques are still available (even inside `lin. system solution via mldivide` or `/`):

- ▶ Pseudo-inverse of a matrix,  $\mathbf{A}^+$ , MATLAB: `pinv(A)`.
- ▶ Least-square-sense solution,  $\min_{\mathbf{x}} \|\mathbf{Ax} - \mathbf{b}\|$ .

In all cases:

- ▶ Be aware or check the condition number, MATLAB: `cond(A)`.

More about [▶ condition number of mathematical operations](#)

## System of Linear Equations – Final Remark (Linearity)



And ... what if we make the following simple change?

$$\begin{bmatrix} 1 & -1 \\ 0.5 & 0.5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & -1 \\ 0.5 & 0.5 \end{bmatrix} \begin{bmatrix} x^2 \\ x^1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$





# System of Linear Equations – Final Remark (Linearity)

And ... what if we make the following simple change?

$$\begin{bmatrix} 1 & -1 \\ 0.5 & 0.5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & -1 \\ 0.5 & 0.5 \end{bmatrix} \begin{bmatrix} x^2 \\ x^1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

No linearity anymore! Entering very different world (still, at least, the solution exist,  $x = 1$ ).

MATLAB:

```
>> syms x
>> solve([x^2 - x == 0, ...
1/2*x^2 + 1/2*x == 1], x)
```

MATHEMATICA:

# Questions?

A8B17CAS

`miloslav.capek@fel.cvut.cz`

September 23

Winter semester 2024/25

---

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.