



**POLITECNICO**  
MILANO 1863

DEPARTMENT  
OF CHEMISTRY MATERIALS  
AND CHEMICAL  
ENGINEERING

# Introduction to MATLAB

Calcoli di Processo dell' Ingegneria Chimica

---

**Timoteo Dinelli**

25<sup>th</sup> of September 2025,

02<sup>nd</sup> of October 2025,

07<sup>th</sup> of October 2025.

Department of Chemistry, Materials and Chemical Engineering, "Giulio Natta", Politecnico di Milano.

**email:** [timoteo.dinelli@polimi.it](mailto:timoteo.dinelli@polimi.it)

## Contacts & Information

**Professor:** Marco Mehl

**e-mail:** marco.mehl@polimi.it

**Office:** Leonardo Campus, Building 6, Department of Chemistry, Materials and Chemical Engineering, “Giulio Natta”, Politecnico di Milano.

**Teacher assistant:** Timoteo Dinelli

**e-mail:** timoteo.dinelli@polimi.it

**Office:** Leonardo Campus, Building 6, Department of Chemistry, Materials and Chemical Engineering, “Giulio Natta”, Politecnico di Milano.

# Resources

## Books:

- ▶ G., Buzzi Ferraris; Manenti, Flavio. Fundamentals and Linear Algebra for the Chemical Engineer: Solving Numerical Problems.
- ▶ G., Buzzi Ferraris; Manenti, Flavio. Interpolation and Regression Models for the Chemical Engineer: Solving Numerical Problems.
- ▶ G., Buzzi Ferraris; Manenti, Flavio. Nonlinear Systems and Optimization for the Chemical Engineer: Solving Numerical Problems.
- ▶ G., Buzzi Ferraris; Manenti, Flavio. Differential and Differential-Algebraic Systems for the Chemical Engineer: Solving Numerical Problems.
- ▶ J. Nathan Kutz. Data-Driven Modeling and Scientific Computation.

- ▶ Steven L. Brunton; J. Nathan Kutz. Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control.
- ▶ A. Quarteroni; R. Sacco; F. Saleri; P. Gervasio. Matematica Numerica.
- ▶ A. Quarteroni; F. Saleri; P. Gervasio. Calcolo Scientifico: Esercizi e problemi risolti con MATLAB e Octave.
- ▶ D. Manca. Calcolo numerico applicato.

### Online Material:

- ▶ Numerical Methods applied to chemical engineering (MIT).
- ▶ GitHub repository of the practical sessions.
- ▶ Matlab online tutorial and documentation.

# Table of Contents

- 1 Introduction
  - What is programming?
  - The Natural Language of Computers
- 2 First Steps with MATLAB
  - Variables
  - Arrays and Matrices
  - While
- 3 Conditional statements
  - If, else if, else

# What is programming?

- ▶ The most simple answer is: “Programming is the act or activity of writing computer programs”. In a more general way programming or coding is the act of writing a computer program.

# What is programming?

- ▶ The most simple answer is: “Programming is the act or activity of writing computer programs”. In a more general way programming or coding is the act of writing a computer program.
- ▶ A computer program is a sequence of instructions that a computer is able to execute.

# What is programming?

- ▶ The most simple answer is: “Programming is the act or activity of writing computer programs”. In a more general way programming or coding is the act of writing a computer program.
- ▶ A computer program is a sequence of instructions that a computer is able to execute.
- ▶ Computer in the definition above is any device that is capable of processing code. This could be smartphones, ATMs, the Raspberry Pi, Servers to name a few.



# What is programming?

- ▶ The most simple answer is: “Programming is the act or activity of writing computer programs”. In a more general way programming or coding is the act of writing a computer program.
- ▶ A computer program is a sequence of instructions that a computer is able to execute.
- ▶ Computer in the definition above is any device that is capable of processing code. This could be smartphones, ATMs, the Raspberry Pi, Servers to name a few.

# What is programming?

- ▶ The most simple answer is: “Programming is the act or activity of writing computer programs”. In a more general way programming or coding is the act of writing a computer program.
- ▶ A computer program is a sequence of instructions that a computer is able to execute.
- ▶ Computer in the definition above is any device that is capable of processing code. This could be smartphones, ATMs, the Raspberry Pi, Servers to name a few.

Remember, every time we use smart devices, some code is running in the background. Moving a mouse pointer from one part of your computer screen to the other may seem like a simple task, but in reality, so many lines of code just ran. An act as simple as typing letters into Google Docs leads to lines of code being executed in the background. It's all code everywhere.

# The Natural Language of Computers

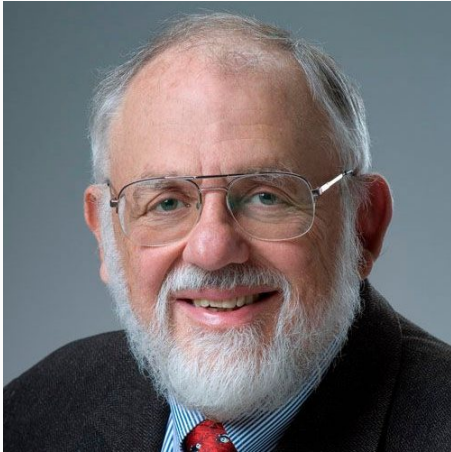
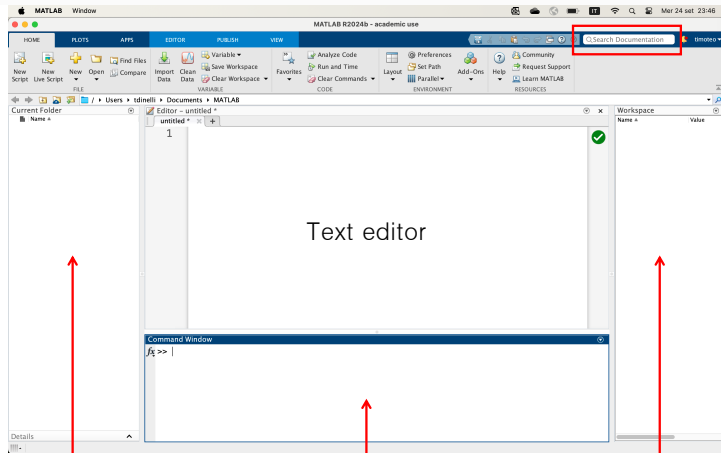


Figure 1: Cleve Barry Moler

The only language understood by a computer is the machine language. A very long list of 0 and 1. However it is a little bit inconvenient to write a series of zeros and ones. So (very smart) people, like the one in the picture, invented what are called programming languages (C/C++, Fortran, python, julia, MATLAB, ...).

**N.B.** Computers aren't very smart, the instructions need to be very precise! Telling a computer what you want it to do is sometimes hard because you have to explain things very carefully and precisely.

# MATLAB



Here is where you will  
find the files

Command Window

Variables

# What can a Computer handle?

- ▶ MATLAB deals with numbers, arrays of numbers, characters, strings.

# What can a Computer handle?

- ▶ MATLAB deals with numbers, arrays of numbers, characters, strings.
- ▶ Numbers can be of different types: signed and unsigned integers, and single-precision and double-precision floating-point numbers.

# What can a Computer handle?

- ▶ MATLAB deals with numbers, arrays of numbers, characters, strings.
- ▶ Numbers can be of different types: signed and unsigned integers, and single-precision and double-precision floating-point numbers.
- ▶ When we set a variable value it gets stored as a series of 0 and 1 in a memory slot whose size depends on the type of variable (i.e., single-precision floating point numbers take 32 bits, or 4 bytes of memory, double precision 64 bit, or 8 bytes).

# What can a Computer handle?

- ▶ MATLAB deals with numbers, arrays of numbers, characters, strings.
- ▶ Numbers can be of different types: signed and unsigned integers, and single-precision and double-precision floating-point numbers.
- ▶ When we set a variable value it gets stored as a series of 0 and 1 in a memory slot whose size depends on the type of variable (i.e., single-precision floating point numbers take 32 bits, or 4 bytes of memory, double precision 64 bit, or 8 bytes).
- ▶ By default, MATLAB store all values as double-precision floating point.



# Area of a cylinder

Let's compute the area of a cylinder given the diameter  $D = 20$  cm and height  $h = 50$  cm.

## Code:

---

```
1 D = 0.2;           % m (diameter)
2 h = 0.5;           % m (height)
3 perimeter = pi * D;
4 Area = perimeter * h;
```

---

## What happens:

1. Create variable **D** = 0.2 m
2. Create variable **h** = 0.5 m
3. Calculate **perimeter** =  $\pi \times 0.2$
4. Calculate **Area** = perimeter  $\times$  0.5

## Source code: scripts and functions in MATLAB

Scripts are m-files (text format) containing MATLAB statements. MATLAB “functions” are another type of m-file. The biggest difference between scripts and functions is that functions have input and output parameters. Script files can only operate on the variables that are “hard-coded” into their m-file.

---

```
1 D = 0.2;           % m
2 h = 0.5;           % m
3 perimeter = pi * D;
4 Area = perimeter * h;
```

---

```
ans = 0.3142
```

---

```
1 function Area = ComputeArea(D, h)
2     perimeter = pi * D;
3     Area = perimeter * h;
4 end
```

---

---

```
ComputeArea(0.2, 0.5)
```

---

```
ans = 0.3142
```

# Variables names

- ▶ MATLAB is case sensitive! **Pippo**  $\neq$  **pippo**
- ▶ Variables name should be self explanatory, so prefer **distance, radius, ...** than **a, b, ...**
- ▶ When possible use the camel case notation to make stuff easier to read.  
**PipeLength, GasTemperature, ...**

# Arrays and Matrices

- In MATLAB **arrays** are defined as:

---

```
v = [5 13 97 31 98]
```

---

v =

5	13	97	31	98
---	----	----	----	----

- **Matrices** can be defined as a set of stacked arrays separated with ;

---

```
M = [5 13 97; 31 98 36; 11 9 20]
```

---

M =

5	13	97
31	98	36
11	9	20

- Elements can be accessed using their index (indices in MATLAB starts from 1)

---

```
v(1)
```

---

```
ans = 5
```

---

```
M(2,1) % (row number, column number)
```

---

```
ans = 31
```

# Creating arrays and matrices

- Create a matrix of zeros or ones:

---

```
A = zeros(3,2)
```

---

A =

0	0
0	0
0	0

---

```
A = ones(2,3)
```

---

A =

1	1	1
1	1	1

- Create a vector of **n** equally spaced elements:

---

```
v1 = [1:2:11]  % Parenthesis can be omitted
```

---

```
v1 =      1      3      5      7      9     11
```

---

```
v2 = [1:3:11]  % Parenthesis can be omitted
```

---

```
v2 =      1      4      7     10  % be careful!
```

---

```
v3 = linspace(1,10,6)  % Parenthesis can NOT be omitted!
```

---

```
v3 =     1.0     2.8     4.6     6.4     8.2    10.0
```

# Operations with arrays and matrices

- Size of a matrix:

---

```
M = [1 2 3; 4 5 6]  
size(M)
```

---

```
ans = 2 3
```

- Copying a matrix:

---

```
A = M
```

---

```
A =  
  
     1     2     3  
     4     5     6
```

- Copying a line or a column of a matrix:

---

```
v = M(1,:)
```

---

```
v =  
  
         1         2         3
```

- Size of an array:

---

```
size(v)
```

---

```
ans = 1 3
```

---

```
length(v)
```

---

```
ans = 3
```



► Matrix transposition:

---

```
C = B'
```

---

```
C =
```

```
     1     4
     2     5
     3     6
```

► Element wise multiplication:

---

```
M .* M
```

---

```
ans =
```

```
     1     4     9
    16    25    36
```

► Matrix multiplication:

---

```
M * B
```

---

Error using \* Inner matrices dimensions must agree.

---

```
M * C
```

---

```
ans =
```

```
    14    32
    32    77
```

N.B. size(M) = 2 3; size(C) = 3 2

## Loops and conditional statements

## for loop

- The **for** loop repeats a series of instructions a **fixed number of times**.

---

```
1 for i = 1:10
2     paperino(i) = i;
3 end
```

---

```
paperino = 1
paperino = 1 2
paperino = 1 2 3
...
paperino = 1 2 3 4 5 6 7 8 9
paperino = 1 2 3 4 5 6 7 8 9 10
```

- The index of the **for** loop can be changed by an arbitrary increment:

---

```
1 for i = 10:-1:1
2     pluto(i) = i;
3 end
```

---

```
pluto = 0 0 0 0 0 0 0 0 0 10
```

```
pluto = 0 0 0 0 0 0 0 0 9 10
```

```
pluto = 0 0 0 0 0 0 0 8 9 10
```

```
...
```

```
pluto = 0 2 3 4 5 6 7 8 9 10
```

```
pluto = 1 2 3 4 5 6 7 8 9 10
```

# Examples

- Sum the first hundred natural numbers.

$$\sum_{i=1}^{100} i = ?.$$

- Sum hundred times one.

$$\sum_{i=1}^{100} 1 = ?$$

---

```
1 sum = 0;  
2 for i = 1:100  
3     sum = sum + i;  
4 end  
5 disp(sum);
```

---

---

```
1 sum = 0;  
2 for i = 1:100  
3     sum += 1;  
4 end  
5 disp(sum);
```

---

## while loop

- ▶ The **while** loop repeats a series of instructions **until a condition is TRUE**.
- ▶ N.B. pay attention with the while loop it can run till infinite, **handle with care**.

---

```
1 sum = 0;
2 while (sum < 325)
3     token = token + 1;
4     sum = sum + token;
5 end
6 disp(['Iteration number: ', num2str(token)]);
7 disp(['Sum is equal to: ', num2str(sum)]);
```

---

Iteration number: 25

Sum is equal to: 325

## IF Statement

- The **IF** statement executes a series of instructions only **IF** a condition is TRUE:

---

```
1 if (condition)
2     % instructions
3 elseif (condition)
4     % instructions
5 elseif (condition)
6     % instructions
7 else
8     % instructions
9 end
```

---

**Example:** Write a simple script to compute the absolute value of a number.

---

```
1 x = 35;
2 if (x >= 0)
3     abs = x;
4 else
5     abs = -x;
6 end
```

---

Thank you for the attention!