CHAPTER 1
Introduction to MATLAB

CHAPTER 1: Introduction to MATLAB

1.1. Introduction

This chapter introduces the fundamental concepts necessary for using MATLAB, a powerful tool for numerical computation and programming. MATLAB is widely used to solve mathematical problems, analyze data, and visualize results, particularly in engineering disciplines. Through this chapter, you will learn to navigate the MATLAB environment, understand the command window, and perform basic operations. Mastery of these concepts is essential as MATLAB forms the foundation for more advanced simulations and modeling tasks.

After studying this chapter, you should be able to:

- Navigate the MATLAB interface and understand its basic components.
- Perform basic arithmetic operations and manage simple data within the MATLAB environment.
- Utilize MATLAB's help resources to enhance your learning and problem-solving abilities.
- Begin developing the skills needed to use MATLAB for more complex engineering applications.

1.2. MATLAB Environment

The word "MATLAB" stands for "MATrix LABoratory." MATLAB is a computer programming language, but you can also use it as a calculator, which is a useful way to experiment with ideas that you might later incorporate into your programs. However, once you move beyond experimentation, you typically rely on MATLAB to create programs that help you perform tasks regularly, easily, and quickly. While these three characteristics don’t encompass everything MATLAB can do, they offer ideas you can pursue and use to your advantage.

The disciplines of Science, Technology, Engineering, and Mathematics (STEM) currently emphasize the importance of simulating mathematical models before experimentation. Innovation in various fields requires these disciplines, as do many practical professions. MATLAB offers a rich and extensive toolbox for STEM that includes statistics, simulation, image processing, symbolic processing, and numerical analysis.

1.2.1. Performing Simple Tasks

Many developers start learning their craft using an older language called Basic (originally written as BASIC). The intention behind Basic was to make the language simple to use.
MATLAB retains the simplicity of Basic but with a much broader toolbox for solving STEM problems. The idea is that you have more important things to do than learn to program in a complex language designed to meet needs that your programs will never address.

Everything comes with trade-offs. MATLAB is specifically designed to meet the needs of those who use mathematics, i.e., modelling physical phenomena. It eliminates the complexity found in many other languages and simplifies things so that you can focus on your work rather than on the tool you use to do it.

However, in seeking simplicity, MATLAB is also less flexible than other programming languages, offering fewer advanced features for tasks you are unlikely to perform and providing fewer general-purpose tools. MATLAB is designed to meet specific needs rather than functioning as a general-purpose language.

1.2.2. Who Uses MATLAB

MATLAB is widely used by professionals whose primary focus is solving real-world problems in their specific fields rather than dealing with the intricacies of computer programming. Its versatility and powerful tools make it popular among scientists, engineers, mathematicians, students, teachers, statisticians, control technology specialists, image processing researchers, and simulation users. These individuals rely on MATLAB to efficiently tackle challenges in their areas of expertise, making it an essential tool across a variety of disciplines.

1.2.3. Why You Need MATLAB

It's crucial to know how to use any application you adopt, but it's equally important to understand when to use it and what it can actually do for your needs. The more a programming language allows you to clearly instruct the computer, the easier it will be to use, and the less time you'll spend getting it to perform a given task.

With the advent of programming languages like C, C++, and Pascal, developers began creating structured environments. In such environments, the flow of instructions and decisions resembles a tree with a trunk and branches, making it much easier to follow and understand.

MATLAB places a strong emphasis on structure—both in how it organizes data and in how you write code. This emphasis means you'll spend more time doing something enjoyable and less time writing code, as the structure allows you to work with data consistently. Early developers could write applications quickly because they had fewer rules to follow. However, newer languages impose structure, making code easier to read and update later, which requires time to learn the rules.

These rules contribute to the learning curve in MATLAB that you need to consider as part of using the product. Be sure to set realistic goals and establish a timeline that reflects the need to learn MATLAB's programming rules. You can't rush the learning process and expect to create something useful at the end.

1.2.4. Utilizing the Powerful Toolbox

MATLAB offers a toolbox designed to meet the specific needs of STEM users. Unlike a general-purpose programming language, this toolbox provides specialized functionalities required to achieve specific objectives.

Below is just a small sample of the areas covered by the tools available in the MATLAB toolbox: Algebra, Linear Algebra—handling numerous equations with many unknowns, Calculus, Differential Equations, Statistics, Curve Fitting, Graphical Representation, and Technical Report Preparation.

1.2.5. What is a Program?

The work of a programmer involves creating computer programs. To do this, the programmer must instruct the computer in a specific language, called a programming language, regarding which data to use and which methods to apply to process this data. A computer program generally performs three tasks:

- **It reads input data:** The program needs to know what it will work with. For example, to use a calculator, you must provide numbers and specify the operations to be performed. Typically, this is done using a keyboard, but the program can also retrieve data from a hard drive or another computer via a network, or other sources.
- **It performs calculations:** Based on the input data, the program automatically applies methods to process the data and produce a result. The methods that computers can execute are called algorithms. For instance, a calculator applies the addition or multiplication algorithm.
- **It writes output data:** Once the program has obtained a result, it must output the result somewhere for use. For example, a calculator displays the result on the screen or stores it in memory.

1.2.6. Key Features and Capabilities of MATLAB

The key features and capabilities of MATLAB include:

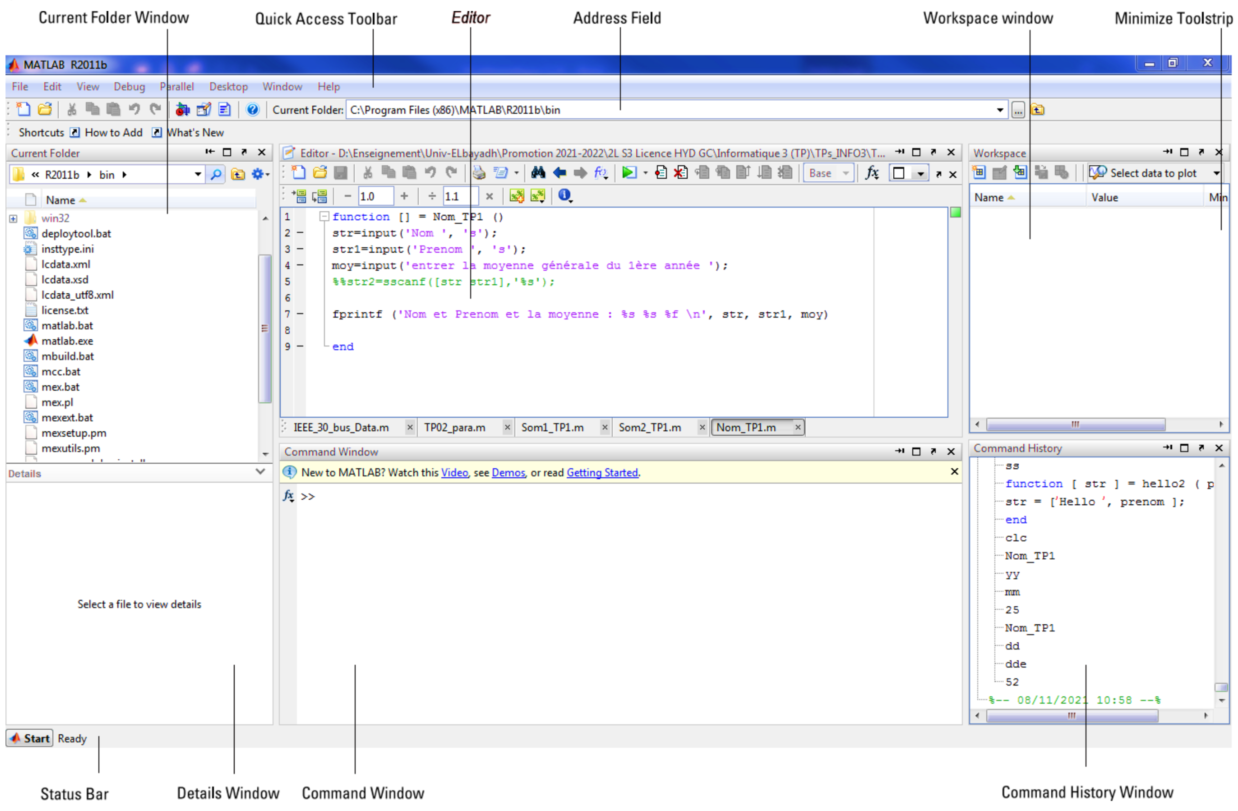
- **Matrix-based computation:** MATLAB is renowned for its efficient handling of matrices and arrays, making it an ideal tool for linear algebra, signal processing, and image processing. Many mathematical operations can be performed directly on entire matrices or vectors, thus simplifying complex calculations.
- **Data analysis and visualization:** MATLAB offers numerous tools for data analysis, statistics, and visualization. Users can generate various types of plots to visualize data, analyse trends, and extract insights from their datasets.
- **Algorithm development:** MATLAB is frequently used for the development and prototyping of algorithms. Its rich library of functions and toolboxes covers a wide range of applications, from optimization and machine learning to control systems and image processing.
- **Interactivity:** MATLAB's interactive environment allows users to execute commands and scripts line by line, facilitating code testing and debugging. This feature is particularly useful for data exploration and experimenting with different algorithms.

- **Cross-platform compatibility:** MATLAB is available on multiple platforms, including Windows, macOS, and Linux, making it accessible to a broad audience of users.
- **Integration:** MATLAB can be integrated with other programming languages such as C/C++, Java, and Python, enabling users to leverage their existing code or interface with external systems.
- **Toolboxes:** MATLAB offers a vast collection of toolboxes, which are specialized libraries designed for specific applications. These toolboxes extend MATLAB's functionality and cover various domains, such as image processing, signal processing, control systems, and more.
- **Simulink:** Simulink is a companion product to MATLAB that provides a graphical environment for modeling, simulating, and analyzing dynamic systems, especially in the context of control systems and physical simulations.
- **Parallel computing and GPU utilization:** MATLAB supports parallel computing and GPU acceleration, allowing users to speed up computationally intensive tasks by leveraging the power of multi-core processors and graphics processing units.
- **Community and support:** MATLAB boasts a robust community of users and developers, and MathWorks provides extensive documentation, tutorials, and customer support resources to help users get the most out of the software.

1.3. MATLAB Windows

Starting with the MATLAB environment aims to familiarize you with the interface and basic windows (workspaces) to use some fundamental functions for reading, displaying, and saving data. We will get acquainted with the MATLAB interface. Depending on the version used, the interface may slightly differ, but the core elements will remain the same.

Figure 1.1 illustrates the MATLAB interface in version 2011.


Current Folder Window
Quick Access Toolbar
Editor
Address Field
Workspace window
Minimize Toolstrip
MATLAB R2011b
File Edit View Debug Parallel Desktop Window Help
Current Folder: C:\Program Files (x86)\MATLAB\R2011b\bin
Shortcuts How to Add What's New
Current Folder
Editor - D:\Enseignement\Univ-ELbayadh\Promotion 2021-2022\2L S3 Licence HYD GC\Informatique 3 (TP)\TPs_INFO3\T...
Workspace
Select data to plot
Name Value Min
IEEE_30_bus_Data.m TP02_para.m Som1_TP1.m Som2_TP1.m Nom_TP1.m
Command Window
Command History
Details
Select a file to view details
Start Ready
Status Bar
Details Window
Command Window
Command History Window

```
function [] = Nom_TP1 ()
str=input('Nom ', 's');
str1=input('Prenom ', 's');
moy=input('entrer la moyenne générale du 1ère année ');
%%str2=sscanf([str str1],'%s');

fprintf ('Nom et Prenom et la moyenne : %s %s %f \n', str, str1, moy)

end
```

```
win32
deploytool.bat
insttype.ini
lcdata.xml
lcdata.xsd
lcdata_utf8.xml
license.txt
matlab.bat
matlab.exe
mbuild.bat
mcc.bat
mex.bat
mex.pl
mexext.bat
mexsetup.pm
mexutils.pm
```

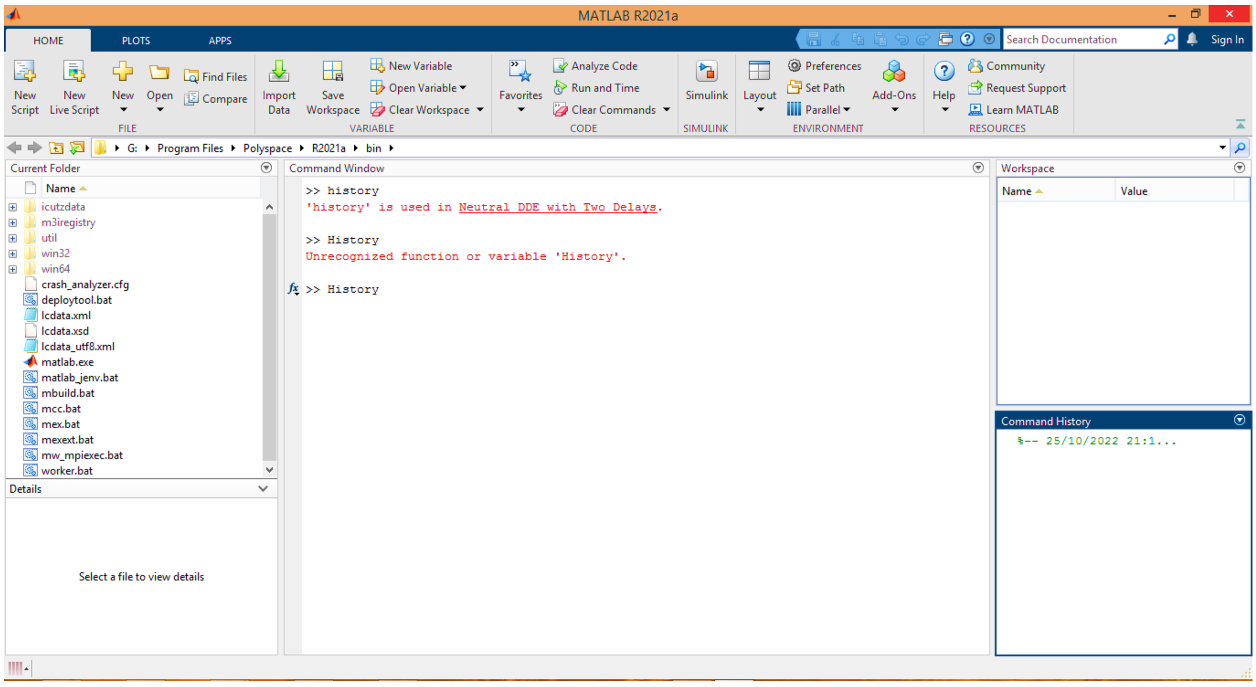
New to MATLAB? Watch this Video, see Demos, or read Getting Started.

```
>>
```

```
ss
function [ str ] = hello2 ( p
str = ['Hello ', prenom ];
end
clc
Nom_TP1
yy
mm
25
Nom_TP1
dd
dde
52
%-- 08/11/2021 10:58 --%
```

Figure 1.1: MATLAB interface, version R2011b

Figure 1.2 shows the MATLAB interface in version R2021a.



```
>> history
'history' is used in Neutral DDE with Two Delays.

>> History
Unrecognized function or variable 'History'.

>> History
```

```
%-- 25/10/2022 21:1...
```

Figure 1.2: MATLAB interface, version R2021a

To display the main MATLAB windows on the user interface, use the menu bar, Quick Access Toolbar >> Desktop (see Figure 1.3).

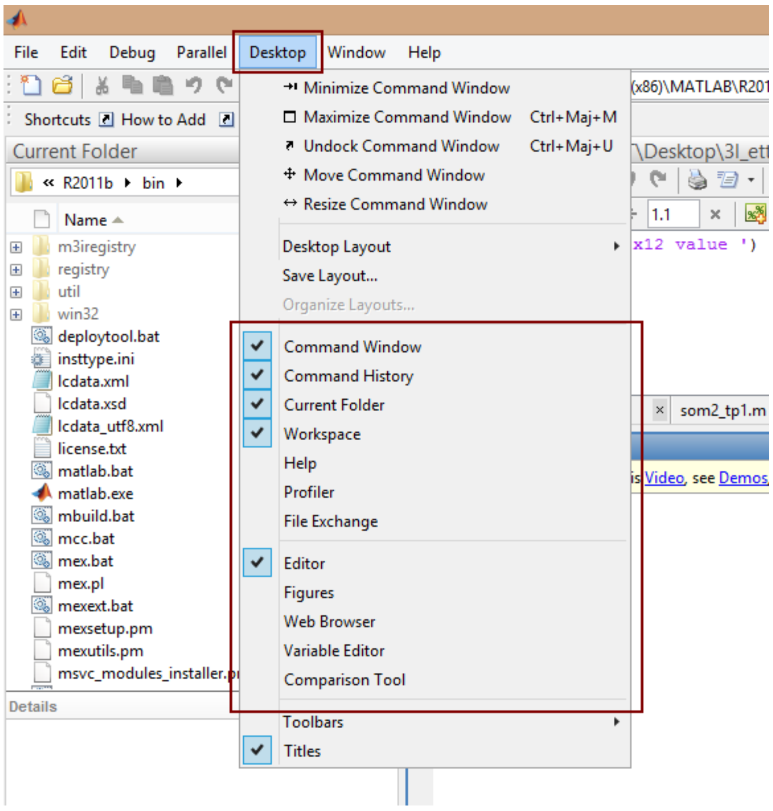


Figure 1.3: Main MATLAB Windows

1.3.1. Command Window

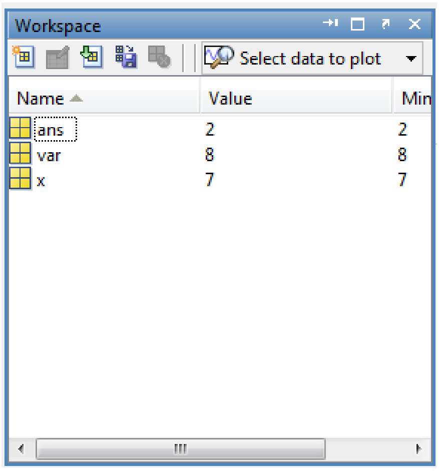
The "Command Window" in MATLAB is one of the main components of its interactive environment. This window is where you directly interact with the software to perform calculations, analyze data, and develop algorithms. It is essential for working effectively within the MATLAB environment, whether for simple tasks or more complex programming projects.

Here’s what you need to know about the Command Window in MATLAB:

- **Interactive Interface:** The Command Window is where you can interact directly with MATLAB by entering commands, mathematical expressions, and scripts. You can type MATLAB commands directly into this window and press "Enter" to execute them.
- **Command Execution:** You can use the Command Window to perform simple calculations, call MATLAB functions, create and manipulate variables, load and analyze data, and much more. Anything you type into the Command Window is immediately evaluated by MATLAB.
- **Displaying Results:** The results of the commands you execute in the Command Window are also displayed in this window. This includes variable values, function outputs, graphs, and error messages if there are issues.
- **Command History:** The Command Window keeps a history of all the commands you have entered during your MATLAB session. You can scroll through this history to reuse or modify previously executed commands.
- **Auto-completion:** MATLAB offers an auto-completion feature in the Command Window. As you start typing a command or variable name, MATLAB will suggest possible options, helping you avoid typing errors.
- **Interactive Environment:** The Command Window is particularly useful for exploring data, testing ideas quickly, and debugging code. You can also use the Command Window to perform simple tasks interactively.
- **Scripts and Functions:** While the Command Window is primarily used for real-time interaction, you can also create and run MATLAB scripts and functions using this window. Simply create a script or function file and then execute it by calling its name in the Command Window.

1.3.2. Workspace Window (*Go to Window -> Workspace*)

In this window, you can see a list of variables known to MATLAB. You can double-click on a variable to display it. A right-click on the variables offers many options such as Copy, Paste, Delete, etc.



Name	Value	Min
ans	2	2
var	8	8
x	7	7

Figure 1.4: Workspace

- **Clear Command:** This command allows you to remove all variables from the Workspace to start with a clean slate.

1.3.3. Current Folder Window (*Go to Window -> Current Folder*)

This is the folder that contains the script files, programming work, and other tasks performed in MATLAB.

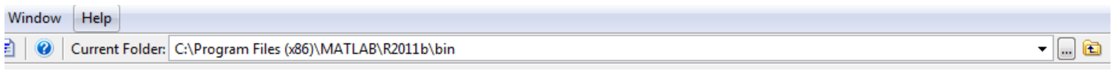
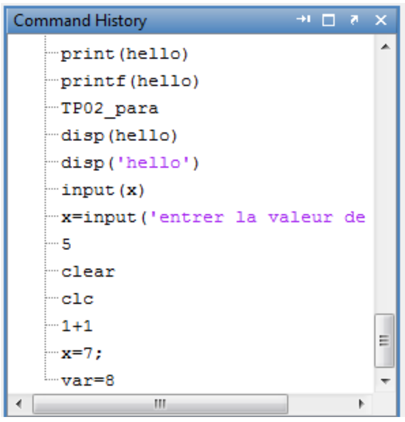


Figure 1.5: Current Folder

1.3.4. Command History Window (*Go to Window -> Command History*)

The Command History window keeps a record of all operations performed in the Command Window. You can also navigate through the command list by placing the cursor in the Command Window and pressing the up and down arrow keys.



```
print(hello)
printf(hello)
TP02_para
disp(hello)
disp('hello')
input(x)
x=input('entrer la valeur de
5
clear
clc
1+1
x=7;
var=8
```

Figure 1.6: Command History

1.4. Overview and General Concepts

1.4.1. Getting Help

The help feature is essential when programming with a high-level language like MATLAB, where the number of functions is vast, and the syntax can be complex. To access help, you can either select a function and press F1 on the keyboard or type in the Command Window: help cos, help input, etc. It is crucial to familiarize yourself with MATLAB's help tools to succeed in this course/lab.

Table 1.1: Help Commands

Command	Description
helpwin	Opens a window containing the list of MATLAB commands along with their documentation
help	Provides a list of all commands by topic
help name	Describes the function name.m
lookfor name	Searches for a command based on the keyword name

On the menu bar, use the window: Product Help → Product Help to display the Help Window.

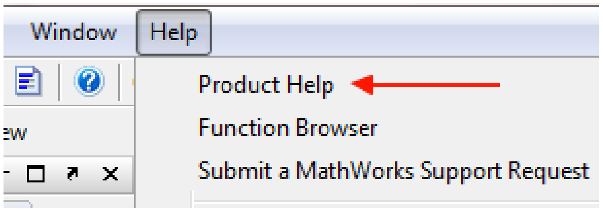


Figure 1.6: Menu Bar → Product Help

1.4.2. Getting Started

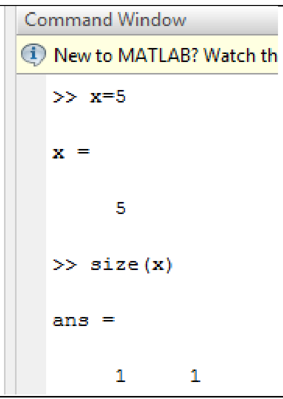
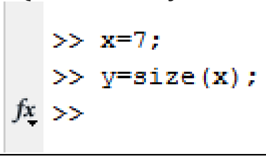
A scalar variable (x=5: assigning the value 5 to the variable x) is viewed by MATLAB as a 1x1 matrix (row x column).

1.4.3. Workspace

This is demonstrated in the following example, where the value 5 is assigned to the variable x, and its dimensions are then queried using the size(x) function in the Command Window:

Table 1.2: Declaration of a Real Variable x

	Using a Semicolon (;)	*ans* (answer)

 Command Window
New to MATLAB? Watch th
`>> x=5`
`x =`
`     5`
`>> size(x)`
`ans =`
`     1     1`
	To prevent the display of results, simply follow the command with a semicolon (;). MATLAB defines a variable ans, which is a 1x1 matrix (one row by one column).  `>> x=7;`
	`>> y=size(x);`
	`>>`
		Matlab defines a variable ans, it is a matrix of size 1x1 (one row by one column).

The *clc* instruction: This command clears the Command Window to provide a clean workspace.

1.4.4. Syntax of a Command Line

In MATLAB, the syntax of a command line is generally simple and straightforward. Here are some key elements to know:

Basic Commands:

- Instructions are written in a line, and commands are executed in the order they appear.
- MATLAB commands are case-sensitive.

On MATLAB, use the following code for better understanding:

1. *a = 5; % Assign the value 5 to variable a*
2. *b = a + 3; % Add 3 to the value of a and assign the result to b*

1.4.4.1. Using a Semicolon (;) to End a Statement

Each command line can be terminated with a semicolon (;). This suppresses the output display in the console.

On MATLAB, use the following code for better understanding:

3. *c = a * b; % Multiply a by b, do not display the result*

1.4.4.2. Script

A script in MATLAB is an .m file that contains a series of MATLAB instructions to be executed. Unlike functions, scripts do not accept input or output arguments, and the variables they use or create are global in MATLAB's workspace.

1.4.4.2.1. Commenting in a Script

Comments are preceded by the % symbol and are not executed by MATLAB. They can be placed on a new line or at the end of a command line.

```
d = 10;  % This is a comment
```

1.4.4.2.2. Multiple Commands on a Single Line

You can also write multiple commands on a single line by separating them with a semicolon (;).
On MATLAB, use the following code for better understanding:

```
e = 2; f = 3; g = e + f;
```

1.4.4.2.3. Line Continuation (Using ...)

To continue a command on the next line, use three dots (...).
On MATLAB, use the following code for better understanding:

```
long_variable_name = 1 + 2 + 3 + ...
                             4 + 5 + 6;
```

1.4.4.2.4. Calling Functions:

Functions can be called directly, and their arguments are passed within parentheses.
On MATLAB, use the following code for better understanding:

```
sqrt_val = sqrt(16);  % Call the sqrt function to calculate the square root of 16
```

1.4.4.2.5. Vector Operations:

MATLAB is particularly powerful for operations on arrays or matrices. These operations can be performed element-wise using a dot (.) before the operator.
On MATLAB, use the following code for better understanding:

```
v = [1, 2, 3, 4];
v_squared = v.^2;  % Squares each element of v
```

This syntax allows you to write MATLAB programs clearly and concisely.

1.4.5. Managing Files in the Working Directory

1.4.5.1. Editor Window

Most of your work in MATLAB will involve creating or modifying files with the .m extension, which defines MATLAB files. When performing a task in MATLAB, it is often possible to do so using only the Command Window. However, when the task becomes more complex (several dozen lines of code) or if you want to share it with someone else easily, you use the Editor Window. An .m file can be created, which can either be a script or a function.

1.4.5.2. MATLAB Function

In MATLAB, a function is defined in a separate file with the .m extension. The file must have the same name as the function.

1.4.6. Arithmetic Operations

1.4.6.1. Example of a MATLAB Function

On MATLAB, use the following code for better understanding.

Imagine you want to create a function that calculates the sum of two numbers. Here's how you can do it:

1. **Creating the File:**
   - Create a new file with the .m extension. For example, name it ***sum.m***.
2. **Defining the Function's Variables (Arguments):** Write the following code in the sum.m script file:

```
function result = sum(a, b)
result = a + b;  % Calculates the sum
end
```

1.4.6.1.1. Code Explanation

- *sum* – The function name takes two inputs and returns their sum.
- **Arguments:**
  - a - First number
  - b - Second number
- **Returns:**
  - result - The sum of variables a and b

1.4.6.1.2. Declaring a MATLAB Function

- The function starts with the keyword ***function***.
- ***"sum"*** is the function's name.
- ***"Result"*** is the output variable that will hold the function's result.
- ***"(a, b)"*** are the function's input arguments (variables).

1.4.6.1.3. Function Body

- ***result = a + b;***: This line performs the function's primary operation, which is the addition of the two numbers a and b.

1.4.6.1.4. End of the Function

- The function automatically ends when there is no more code to execute. It is not necessary to explicitly use the end command for simple functions, but it is recommended

1.4.7. Operations and Functions Using the somme(a, b) Function

Once you have created and saved the file somme.m, you can call this function from the MATLAB command line or another script as follows:

```
x = 3;
y = 5;
z = somme(x, y); % Calls the somme function with x and y as arguments
disp(z); % Displays the result
```

This will display 8, which is the sum of 3 and 5. The command disp(z) outputs the content of the variable z.

Part 02: Simulation Section (MATLAB - SIMULINK)

To start, we set the USERPATH, the folder containing your (.m) file.

1. Create a folder on your Desktop named: TP_INFO3.
2. Inside the previous folder, create another folder named: TP01_INFO3.
3. In the Current Folder space as shown in figure 5, select the TP01_INFO3 folder for the first lab session, and similarly for subsequent lab sessions.

Exercise 1: "Hello World"

In this exercise, you will create a simple program to display the famous "Hello World" message using MATLAB. This exercise serves as an introduction to writing and executing basic code in MATLAB in script mode.

Here's the basic code snippet for displaying "Hello World" in MATLAB:

```
>>disp('Hello World');
```

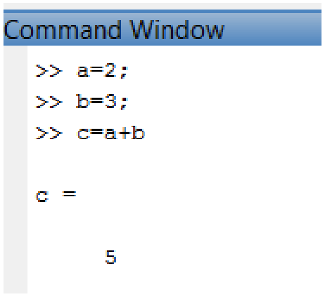
You can create a new script file (.m) in the designated folder, for example, TP01_INFO3, and add the above code to it.

Then, run ▶ the script in MATLAB to see the "Hello World" message displayed in the **Command Window**.

Exercise 2: "Sum of Two Numbers 2 and 3"

To calculate the sum of the numbers 2 and 3 using the function **Sum_TP1**, you can follow these steps:

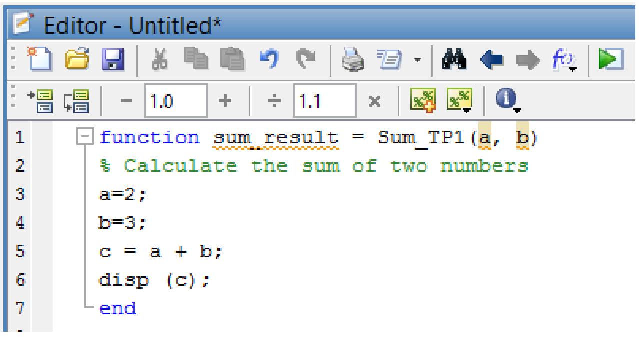
1. In the Command Window, use this script:



```
>> a=2;
>> b=3;
>> c=a+b

c =

     5
```

Figure 1. Script Sum of Two Numbers 2 and 3

2. In the Editor Window, you can use the function with the numbers 2 and 3 as arguments:



```
function sum_result = Sum_TP1(a, b)
% Calculate the sum of two numbers
a=2;
b=3;
c = a + b;
disp (c);
end
```

Figure 2. Function Sum of Two Numbers 2 and 3

Exercise 3: "Sum of Any Two Numbers"

Create a new script file in the designated folder, TP01_INFO3, and add the above code. When you run the script in MATLAB, it will prompt you to input two numbers and then display the sum of those numbers in the Command Window.

Let's attempt to create a script and then convert it into a function that takes two numbers as input and returns their sum [2.3].

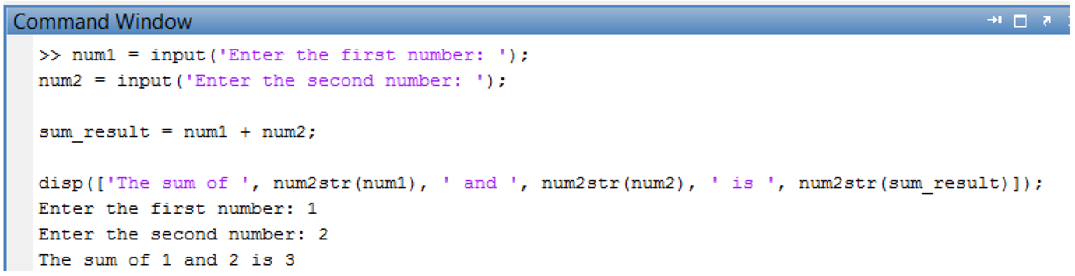
Follow these steps:

1. Create a new script file (Ctrl + N).
2. Write a program that calculates the sum of two numbers; numbers x and y.
3. Save the file with the name: **Sum_xy_TP1.m** (use underscore "_ ").

Here's how you can structure your script and then convert it into a function:

With Script Version (Sum_xy_TP1.m):

1. *>>% Calculate the sum of two numbers x and y*
2. *>>x = input('Enter the first number: ');*
3. *>>y = input('Enter the second number: ');*
4. *>>sum_result = num1 + num2;*
5. *>>disp(['The sum of ', num2str(x), ' and ', num2str(y), ' is ', num2str(sum_result)]);*



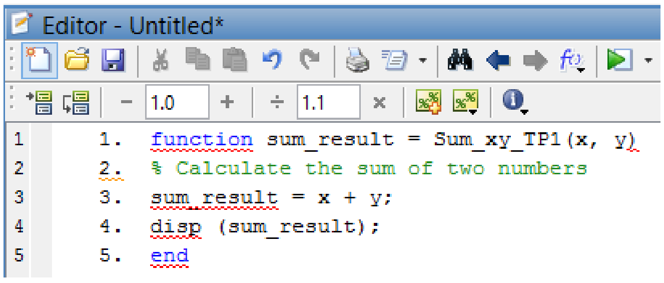
Command Window

```
>> num1 = input('Enter the first number: ');
num2 = input('Enter the second number: ');

sum_result = num1 + num2;

disp(['The sum of ', num2str(num1), ' and ', num2str(num2), ' is ', num2str(sum_result)]);
Enter the first number: 1
Enter the second number: 2
The sum of 1 and 2 is 3
```

*Figure 3. Exercise 2, Sum of Two Numbers: **Script Version** (Sum_xy_TP1.m)*

With Function Version (Sum_xy_TP1.m):

1. *function sum_result = Sum_xy_TP1(x, y)*
2. *% Calculate the sum of two numbers*
3. *sum_result = x + y;*
4. *disp (sum_result);*
5. *end*



Editor - Untitled*

```
1.  function sum_result = Sum_xy_TP1(x, y)
2.  % Calculate the sum of two numbers
3.  sum_result = x + y;
4.  disp (sum_result);
5.  end
```

*Figure 4. Exercise 2, Sum of Two Numbers: **Function Version** (Sum_xy_TP1.m)*

In the function version, you define a function named Sum_xy_TP1 that takes two input arguments x and y, calculates their sum, and returns the result. Remember to save the function file in the same folder, and you can then call this function from the Command Window or other scripts to calculate the sum of two numbers when you need.

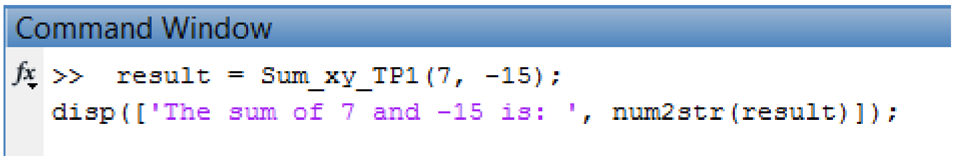
Exercise 4: "Testing the function Sum_xy_TP1.m"

In this exercise, you'll create a MATLAB function that calculates the sum of the two numbers using the function **Som_xy_TP1** that you've created, this exercise will reinforce your understanding of creating functions in MATLAB, you can follow these steps.

1. Make sure you have the Sum_xy_TP1 function defined in a file named "Sum_xy_TP1.m" in your MATLAB working directory **(Current Folder).**
2. In the Command Window, you can call the function with any two numbers as arguments:

1. >>result = Sum_xy_TP1(2, 3);

2. >>disp(['The sum of 2 and 3 is: ', num2str(result)]);



```
Command Window
>>  result = Sum_xy_TP1(7, -15);
disp(['The sum of 7 and -15 is: ', num2str(result)]);
```

Figure 5. Call the function Sum_xy_TP1.m

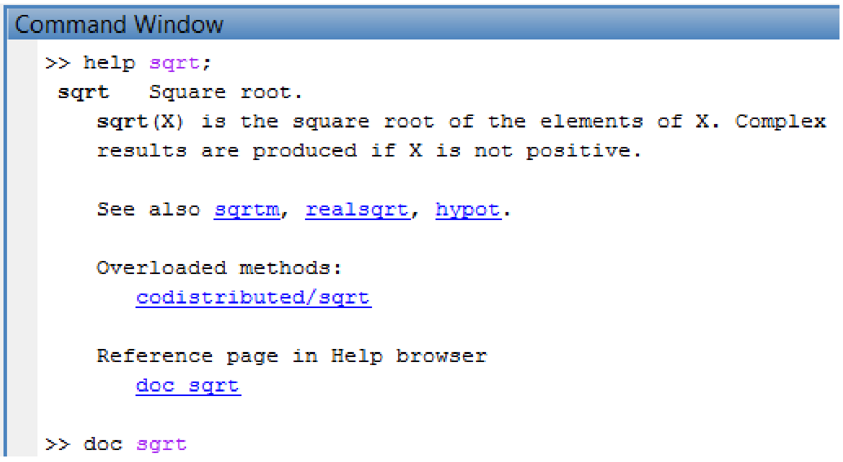
This demonstrates how you can use the custom function Sum_xy_TP1 to calculate the sum of any two numbers.

Exercise 5: "How to Find Desired Information"

In this exercise, you'll practice using MATLAB's **help** and documentation resources to find the information you need. These skills are crucial for effectively using MATLAB's capabilities and functions. Follow these steps:

1. Go to the **Command Window** in MATLAB.
2. Write **help** command or the **doc** command to search for information about a specific function or topic.

For example, let's say you want to find information about the **sqrt** function, which calculates the square root of a number:



```
Command Window
>> help sqrt;
 sqrt   Square root.
    sqrt(X) is the square root of the elements of X. Complex
    results are produced if X is not positive.

    See also sqrtm, realsqrt, hypot.

    Overloaded methods:
       codistributed/sqrt

    Reference page in Help browser
       doc sqrt

>> doc sqrt
```

Figure 6. Find information about the sqrt function

This command (doc sqrt) will display the help documentation for the sqrt function, including its syntax, description, and usage examples. Similarly, you can use the help or doc command to explore other functions and topics within MATLAB to gain a deeper understanding of their functionalities and how to use them. Remember, being able to effectively search for and use documentation is a valuable skill for programming in MATLAB.

Part 03: Experimental Section (MATLAB - SIMULINK)

Exercise 6: "Calculate operations"

In this exercise, you'll create a MATLAB program that takes three numbers as input and displays the product of the first two numbers and the square root of the product of the last two numbers. This exercise will reinforce your understanding of basic calculations and functions in MATLAB. Follow these steps:

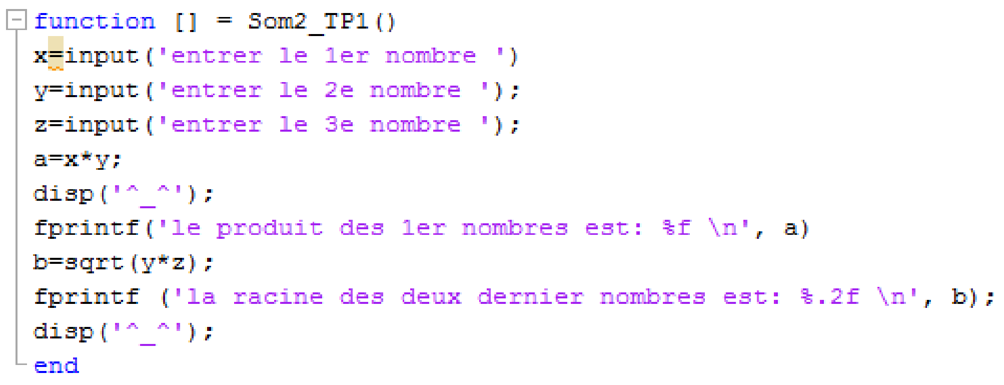
1. Open a new script file (Ctrl + N).
2. Write a program that takes three numbers as input, calculates the desired values, and displays the results.
3. Save the file with an appropriate name, such as "Sum_Product_Root.m".

Here's how you can structure your script for this exercise:

1. *>>% Input three numbers*
2. *>>num1 = input('Enter the first number: ');*
3. *>>num2 = input('Enter the second number: ');*
4. *>>num3 = input('Enter the third number: ');*
5. *>>% Calculate product of the first two numbers*
6. *>>product_result = num1 * num2;*
7. *>>% Calculate square root of product of the last two numbers*
8. *>>root_result = sqrt(num2 * num3);*
9. *>>% Display the results*
10. *>>disp(['The product of ', num2str(num1), ' and ', num2str(num2), ' is ', num2str(product_result)]);*
11. *>>disp(['The square root of the product of ', num2str(num2), ' and ', num2str(num3), ' is ', num2str(root_result)]);*

Create a new script file, add the above code, and run the script in MATLAB. It will prompt you to input three numbers and then display the calculated values as described.

Using this function in Editor Window:



```
function [] = Som2_TP1()
x=input('entrer le 1er nombre ')
y=input('entrer le 2e nombre ');
z=input('entrer le 3e nombre ');
a=x*y;
disp('^_^');
fprintf('le produit des 1er nombres est: %f \n', a)
b=sqrt(y*z);
fprintf ('la racine des deux dernier nombres est: %.2f \n', b);
disp('^_^');
end
```

Figure 7. Calculate operations

In this script we used 4 functions "**input(), sqrt(), dips() and fprintf()**",

- Use the command "help" to discover these functions.

- **What is the purpose of these functions?**

Exercise 7: "Enter First and Last Name"

In this exercise, you'll create a MATLAB function that takes your first name and last name as input arguments and returns them on the same line. This exercise will help you practice creating and using functions with input and fprintf command in MATLAB.

Follow these steps:

1. Create a new script file (Ctrl + N).
2. Write a program in Editor that defines a function to concatenate your first name and last name and display them.
3. Save the file with an appropriate name, such as "Name_TP1.m".

Here's how you can structure your script to create the function:

Script Version (FullNameConcatenation.m):

```
% Concatenate first name and last name
F=input(‘enter your first name’);
L= input(‘enter your last name’);
fprintf(‘ full_name is : %s, %s ‘, F, L);
```

Function Version (FullNameConcatenation.m):

```
function Name_TP1 = Name_TP1(first_name, last_name)
% Concatenate first name and last name
full_name = [first_name, ' ', last_name];
end
```

- In this function, you define a function named *Name_TP1* that takes two input arguments first_name and last_name, concatenates them with a space in between, and returns the full name.
- Now, save the function file in the same folder and call this function with your first and last name as arguments to see your full name displayed on the same line.

For example, in the Command Window:

```
>> result = Name_TP1(Abdelkader, Amir);
>> disp(['Your full name is: ', result]);
result =
Abdelkader Amir
```

Exercise 8: "Enter First and Last Name + First-Year Overall Average"

In this exercise, you'll create a MATLAB function that takes your first name, last name, and your first-year overall average as input arguments and returns them on the same line. This exercise will help you practice creating functions with multiple input arguments in MATLAB.

Follow these steps:

1. Create a new script file (Ctrl + N).
2. Write a program that defines a function to concatenate your full name and your first-year overall average and then display them.
3. Save the file with the name: "Name_Moy_TP1.m".

1.5. Conclusion

This chapter has introduced you to MATLAB, its environment, and the fundamental operations necessary to begin using this powerful tool. By understanding the basics of MATLAB, you are now equipped to perform essential tasks such as data management, executing commands, and navigating the interface. These skills provide the foundation for more advanced topics in simulation and modeling, which will be explored in the following chapters. Mastery of MATLAB is crucial for any engineer or scientist working with computational tools.