University mohamed boudiaf of m'sila

Faculty of Mathematics and Computer Science

Department of Mathematics

Domain: Mathematics and Computer Science

Field: Mathematics

Level: math license second year

Polycopied course for the module

Programming Tools 2 with MATLAB

Mihoubi Hamza

E-mail: hamza.mihoubi@univ-msila.dz

Year 2021-2022

Table of Contents

First part: Basic elements

Introduction

This document is an introduction to MATLAB, scientific computing software. Its objective is to prepare the student for practical work in Automatic Control, Mechanics and Numerical Analysis in which this tool is intensively used for the application and simulation of the theoretical principles presented in class. In addition, this manual offers the opportunity for the student to train in widely used professional software.

Cleve Moler, then a professor of computer science at the University of New Mexico, created MATLAB in the 1970s to help his students. It was engineer Jack little who identified the commercial potential of MATLAB in 1983. C. Moler, J. little, and Steve Bangart created MathWorks in 1984 and rewrote MATLAB in C.

MATLAB allows interactive work either in command mode or in programming mode;

While still having the possibility of making graphic visualizations. Considered one of the best programming languages (C or FORTRAN), MATLAB has the particularities following with respect to these languages:

- Easy programming,
- Continuity among integer, real and complex values,
- The wide range of numbers and their precision,
- The very comprehensive mathematical library,
- The graphical tool which includes graphical interface functions and utilities,
- The possibility of linking with other classic programming languages (C or FORTRAN).

The best way to learn how to use this software is to use it yourself, by experimenting, making mistakes and trying to understand the error messages that will be returned to you. These messages are in English! This document is intended to help you for some first steps with MATLAB.

First part: Basic elements

1.1 Interface MATLAB

1.1.1 Command Window

1.1.2 Boutton Start

1.1.3 Workspace

1.1.4 Current Directory

1.1.5 Command History

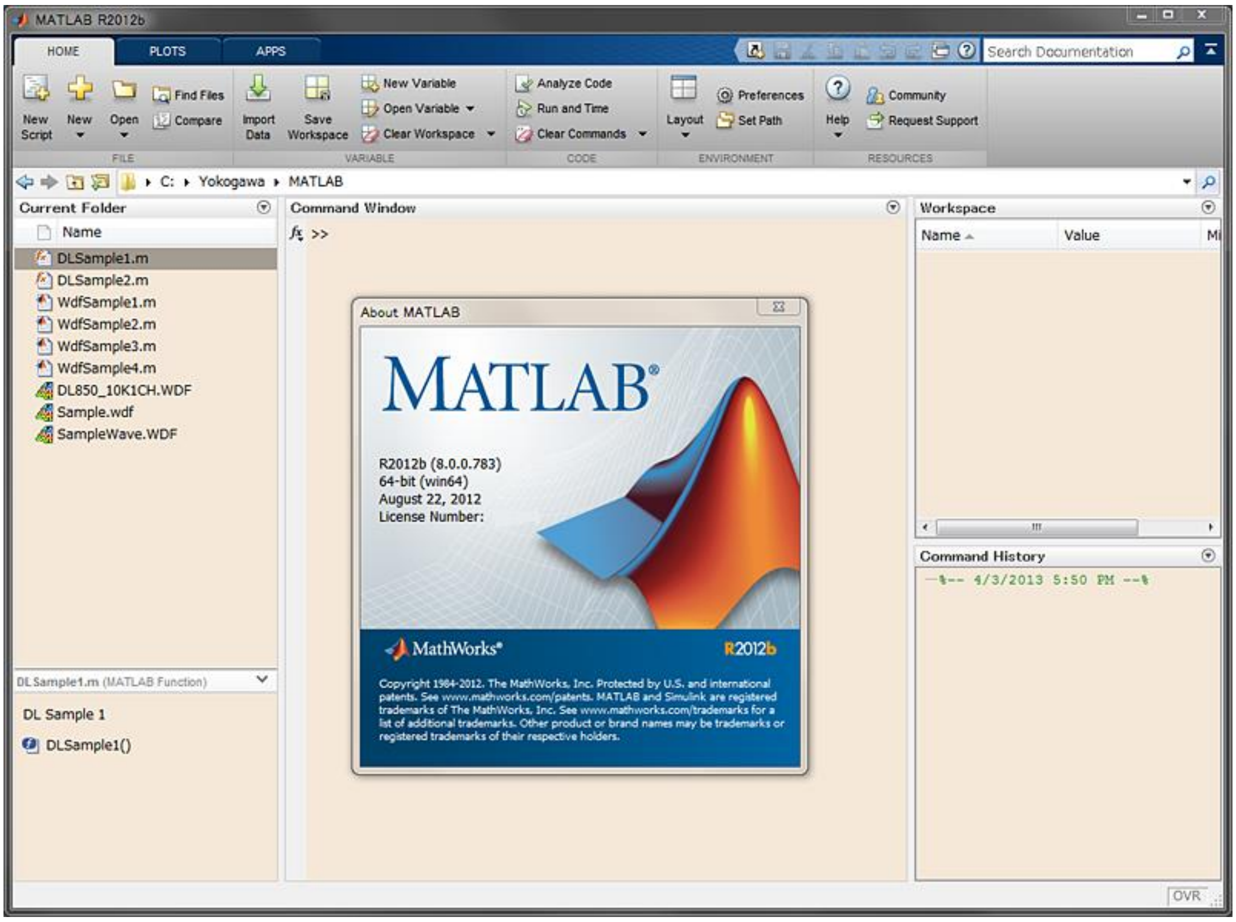
1.1.6 Editor/Debugger

1.1.7 The M-files

First part: Basic elements

1.1 Interface MATLAB

When MATLAB is launched, the following interface appears:



```
R2012b (8.0.0.783)
64-bit (win64)
August 22, 2012
License Number:
```

Copyright 1984-2012. The MathWorks, Inc. Protected by U.S. and international patents. See www.mathworks.com/patents. MATLAB and Simulink are registered trademarks of The MathWorks, Inc. See www.mathworks.com/trademarks for a list of additional trademarks. Other product or brand names may be trademarks or registered trademarks of their respective holders.

```
--%-- 4/3/2013 5:50 PM --%
```

There are three windows. The upper left window therefore displays Workspace. Below is the current directory and the Command History.

Finally, on the right, there is the Command Window. To "close" an interface window, simply click on the button representing an arrow in the upper right corner of each window.

The interface also has menu bars [1].

The commands available in the menus are:

Edit! Clear Command Window	Clear instructions or results can be seen in the Command Window.
Edit! Clear Command History	Cancel previously recorded commands.
Edit! Clear Command Workspace	Erases stored variables from memory.
View	Controls the visual aspects of different windows.

The status bar also contains buttons that allow you to easily add, modify or remove acknowledgment requests and comments.

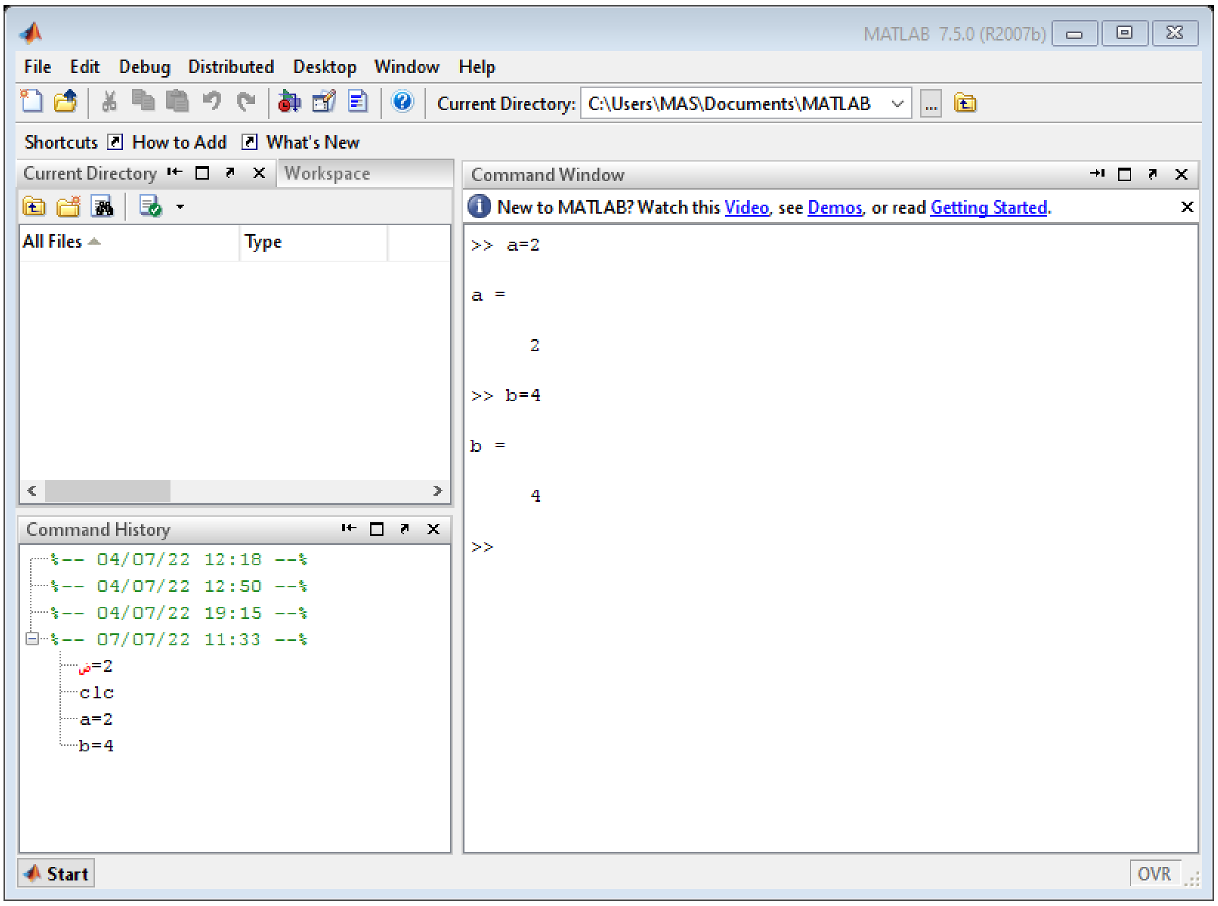
The Current Directory can be determined by this bar without having to go through the window of the same name.

By clicking slipping the horizontal and vertical bars separating the different windows of interface, it is possible to extend or to make look smaller these windows.

First part: Basic elements

1.1.1 Command Window

One of the most important windows of MATLAB, Command Window treats given instructions. It is after invitation ('prompt ') » that it is necessary to enter asked instructions. Results will be displayed from the return of line.


MATLAB 7.5.0 (R2007b)
File Edit Debug Distributed Desktop Window Help
Current Directory: C:\Users\MAS\Documents\MATLAB
Shortcuts How to Add What's New
Current Directory
Workspace
All Files
Type
Command Window
New to MATLAB? Watch this Video, see Demos, or read Getting Started.
Command History
Start
OVR

```
>> a=2

a =

     2

>> b=4

b =

     4

>>
```

```
%-- 04/07/22 12:18 --%
%-- 04/07/22 12:50 --%
%-- 04/07/22 19:15 --%
%-- 07/07/22 11:33 --%
  ض=2
  clc
  a=2
  b=4
```

You can avoid retaking an instruction by hitting the up arrow ("), which will display the previous instruction. Continue to weigh until the desired instruction.

It is easy to observe on the image the instructions given as well as the answers obtained:

Display Instructions

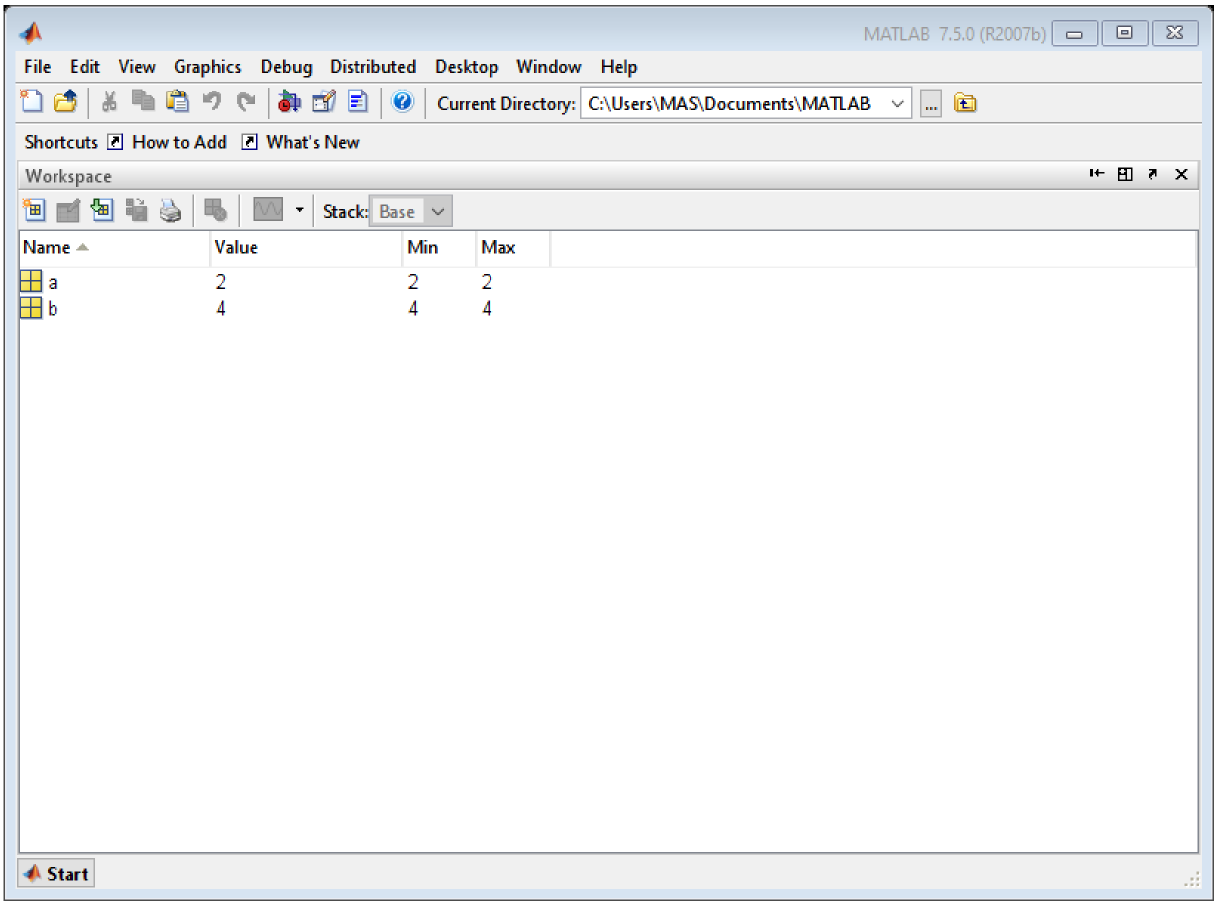
```
>> a=2
a =
2
>> 2*a
ans =
4
>> x1=2*a+6
x1 =
10
>> a*b
ans =
8
```

1.1.2 Boutton Start

The Start button is a tool to quickly open certain MATLAB functions. For more information, see its topic in MATLAB Help.

1.1.3 Workspace

The Workspace window allows you to view stored variables. It contains their name, dimensions and the type of variable. Since Matlab is based on matrices, all the variables are made up of several dimensions: a scalar is a 1 1 matrix and a vector is a 1 n or n 1 matrix, etc. It is possible to delete some variables as well as to edit them. To clear them all, use the Clear Workspace command in the Edit menu.


MATLAB 7.5.0 (R2007b)
File Edit View Graphics Debug Distributed Desktop Window Help
Current Directory: C:\Users\MAS\Documents\MATLAB
Shortcuts How to Add What's New
Workspace
Stack: Base
Start

Name	Value	Min	Max
a	2	2	2
b	4	4	4

By double-clicking on a variable, the Array Editor window appears. This window contains the values of the variables and allows you to modify them. In the following example, the initial variable is a 1 1 matrix. By adding values in the adjacent boxes, the matrix was transformed to 5 3 dimensions.

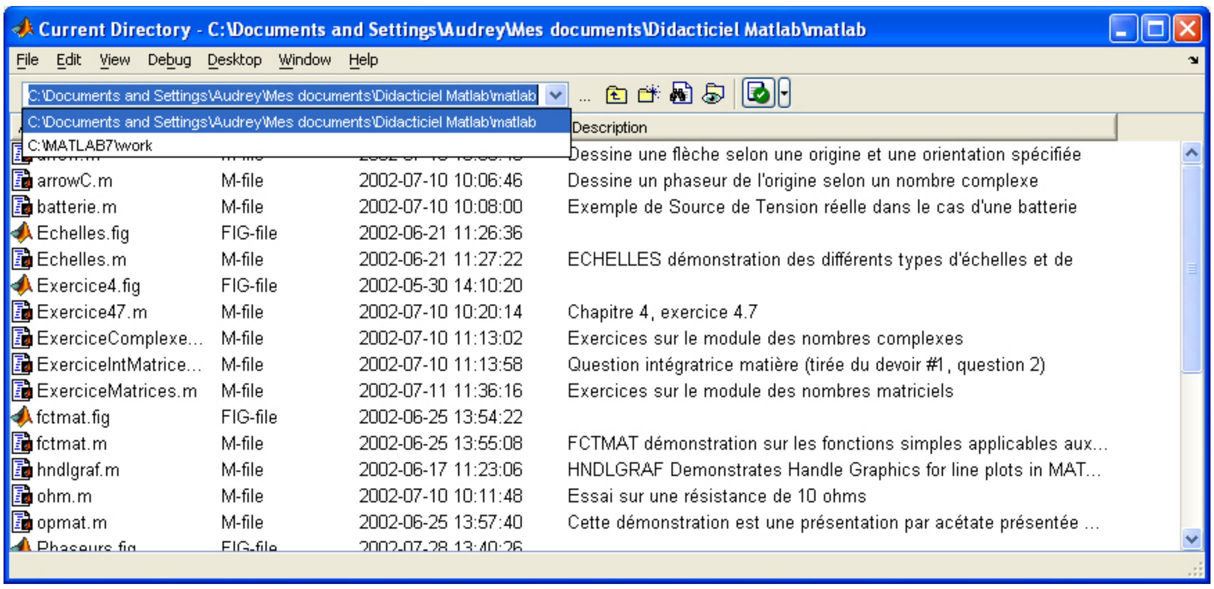
1.1.4 Current Directory

Current Directory is the common directory where are recorded files-Mr. It is hard recommended to create a directory other than that provides by Matlab to manage better files contained indoors. Although you will learn it in the course of educational software program, some general principles in comparison with Current Directory are necessary:

To compile an M-file, it must be saved in the current directory.

If an M-file calls a function other than a Matlab function (i.e. a function created by the user), the M-file calling the function and the M-file defining the function must be in the same current directory.

First part: Basic elements

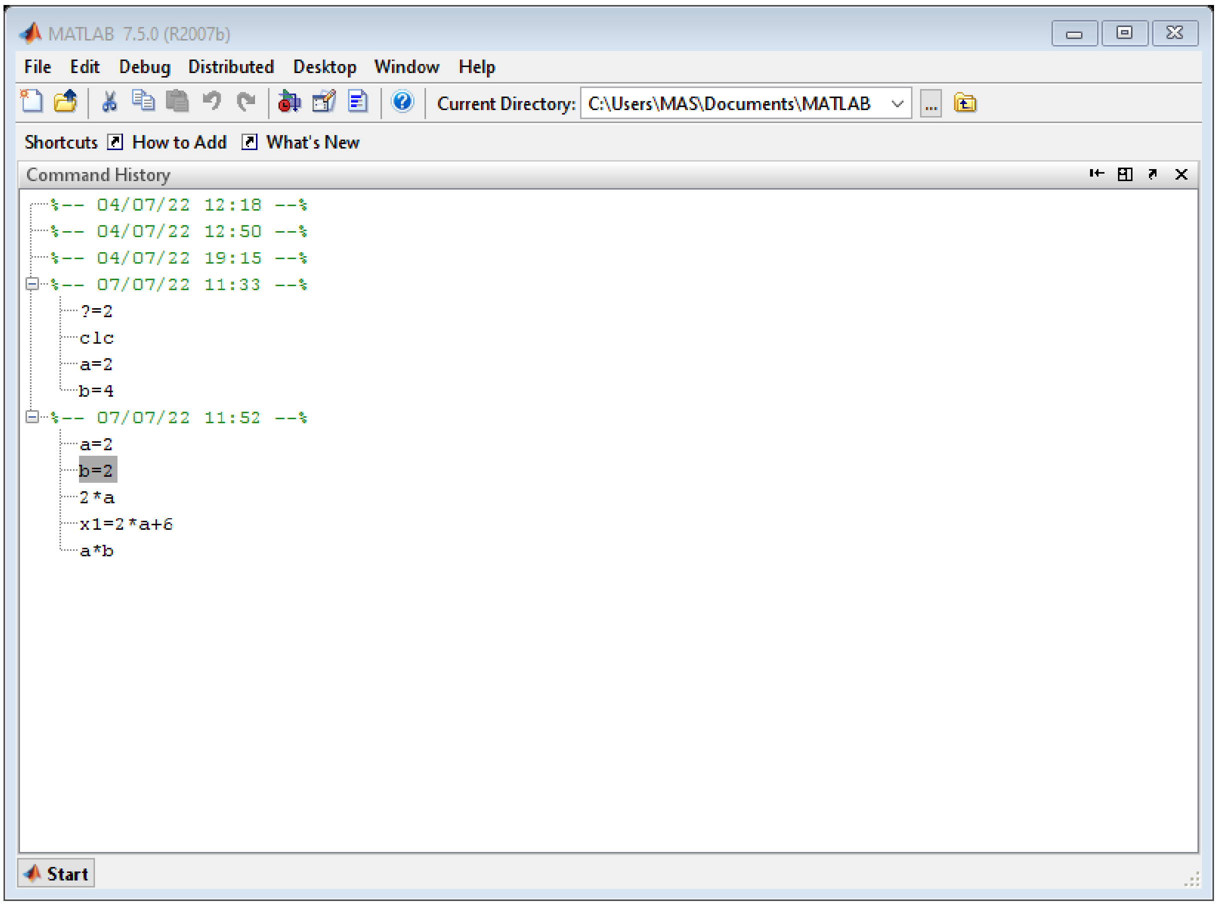


Current Directory - C:\Documents and Settings\Audrey\Mes documents\Didacticiel Matlab\matlab
File Edit View Debug Desktop Window Help
C:\Documents and Settings\Audrey\Mes documents\Didacticiel Matlab\matlab
C:\Documents and Settings\Audrey\Mes documents\Didacticiel Matlab\matlab
C:\MATLAB7\work

			Description
			Dessine une flèche selon une origine et une orientation spécifiée
arrowC.m	M-file	2002-07-10 10:06:46	Dessine un phaseur de l'origine selon un nombre complexe
batterie.m	M-file	2002-07-10 10:08:00	Exemple de Source de Tension réelle dans le cas d'une batterie
Echelles.fig	FIG-file	2002-06-21 11:26:36	
Echelles.m	M-file	2002-06-21 11:27:22	ECHELLES démonstration des différents types d'échelles et de
Exercice4.fig	FIG-file	2002-05-30 14:10:20	
Exercice47.m	M-file	2002-07-10 10:20:14	Chapitre 4, exercice 4.7
ExerciceComplexe...	M-file	2002-07-10 11:13:02	Exercices sur le module des nombres complexes
ExerciceIntMatrice...	M-file	2002-07-10 11:13:58	Question intégratrice matière (tirée du devoir #1, question 2)
ExerciceMatrices.m	M-file	2002-07-11 11:36:16	Exercices sur le module des nombres matriciels
fctmat.fig	FIG-file	2002-06-25 13:54:22	
fctmat.m	M-file	2002-06-25 13:55:08	FCTMAT démonstration sur les fonctions simples applicables aux...
hndlgraf.m	M-file	2002-06-17 11:23:06	HNDLGRAF Demonstrates Handle Graphics for line plots in MAT...
ohm.m	M-file	2002-07-10 10:11:48	Essai sur une résistance de 10 ohms
opmat.m	M-file	2002-06-25 13:57:40	Cette démonstration est une présentation par acétate présentée ...
Phaseurs.fig	FIG-file	2002-07-28 13:40:26	

1.1.5 Command History

Command History inscribes orders as they are appellées in Command Window. It keeps these orders in memory, as well as the date and the opening time of each of the sessions of Matlab.



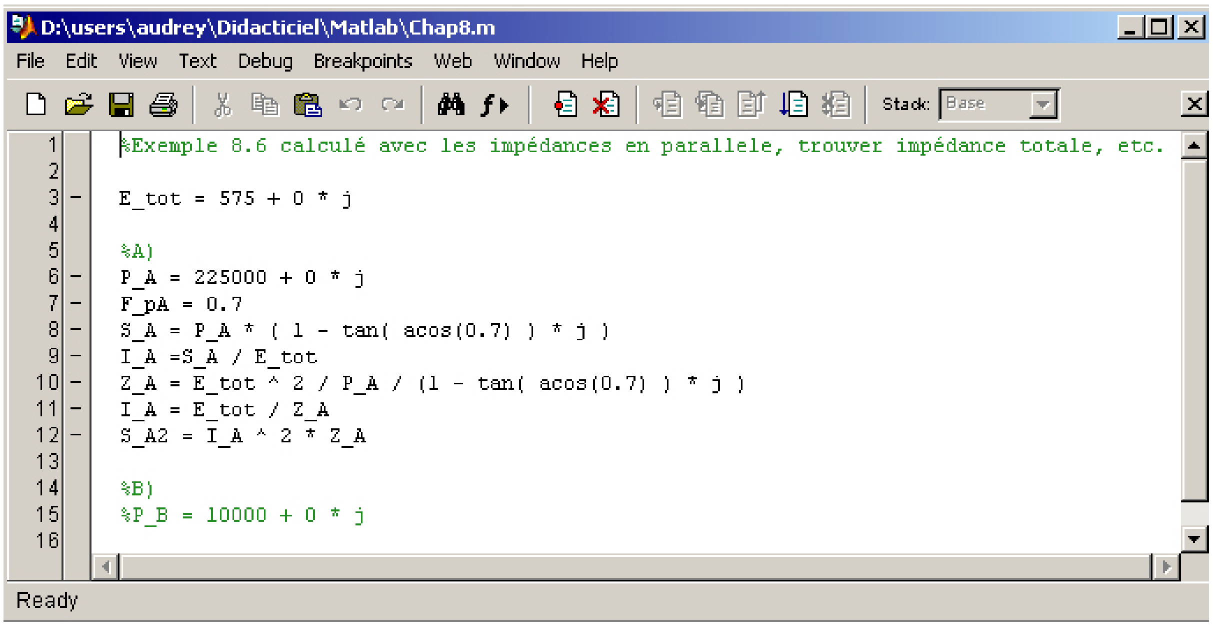
MATLAB 7.5.0 (R2007b)
File Edit Debug Distributed Desktop Window Help
Current Directory: C:\Users\MAS\Documents\MATLAB
Shortcuts How to Add What's New
Command History
Start

```
%-- 04/07/22 12:18 --%
%-- 04/07/22 12:50 --%
%-- 04/07/22 19:15 --%
%-- 07/07/22 11:33 --%
   ?=2
   clc
   a=2
   b=4
%-- 07/07/22 11:52 --%
   a=2
   b=2
   2*a
   x1=2*a+6
   a*b
```

1.1.6 Editor/Debugger

To process M-files, you must use the Matlab editor/debugger. This window is not part of the basic Matlab interface and opens when you open an M-file or when you create a new M-file.

First part: Basic elements


D:\users\audrey\Didacticiel\Matlab\Chap8.m
File Edit View Text Debug Breakpoints Web Window Help
Stack: Base
Ready

```
%Exemple 8.6 calculé avec les impédances en parallele, trouver impédance totale, etc.

E_tot = 575 + 0 * j

%A)
P_A = 225000 + 0 * j
F_pA = 0.7
S_A = P_A * ( 1 - tan( acos(0.7) ) * j )
I_A =S_A / E_tot
Z_A = E_tot ^ 2 / P_A / (1 - tan( acos(0.7) ) * j )
I_A = E_tot / Z_A
S_A2 = I_A ^ 2 * Z_A

%B)
%P_B = 10000 + 0 * j
```

On the bar of buttons, a button is essential: the button RUN compiles program, i.e. carry out the orders of program. He can also be carried out with F5.

1.1.7 The M-files

In order to avoid having to retype a series of commands, it is possible to create a MATLAB program, known as an “M-file” (“M-file”), the name coming from the ending “.m of these files. Using the MATLAB editor, create a text file that contains a series of MATLAB commands. To create an M-file, go to the File New M-File menu or click the blank button. Recording is normally done in the current directory. Once the file has been saved (under the filename.m for example), it is necessary to call it in MATLAB using the command:

>> filename

The commands stored there will then be executed and the results displayed in the Command Window. If you need to make a change to your series of commands, just edit the line of the M-file in question and run the M-file by entering the file name in MATLAB again.

Other ways to run the M-file is to click the Run button, or go to the Debug Run menu in the Editor/Debugger, or press F5.

M-files save you from having to retype a series of repeated commands and allow you to preserve your instructions, commands and calculations thanks to the recording. This is the recommended procédure for your labs.

Example

Creating an M-File - Procedure

Create a file by the button or menu

Write some instructions. For example, write the following instructions:

```
A=90
B=100
C=A+B
```

– Save the m-file in the current directory. If you don't, Matlab will ask you to save it before compiling it.

– Return to Matlab: the responses should have appeared in the command window. These answers should be :

```
>> A =
90
B =
100
C =
190
```

- **The ";" and the "..."**

The semicolon at the end of a line tells MATLAB not to return the result of the operation to the screen. A common practice is to put “;” at the end of all the lines and remove some of them when something goes wrong in our program, in order to see what happens.
In the editor/debugger as in the command window, the use of «. .» is useful to continue on the line below the current instruction.

```
>> B = pi * 3^2
B =
28.274
>> B = pi * 3^2;
>>
```

The following two instructions give the same result

```
>> A = 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9
A =
45
>> A = 1 + 2 + 3 + 4 + 5...
+ 6 + 7 + 8 + 9
A =
45
```

1.2. Use of Matlab in the manner of a scientific calculator

First part: Basic elements

1.2 Use of MATLAB in the manner of a scientific calculator

1.2.1 Special variables and constants

ans	**the most recent answer**
pi	number pi
inf	plus infinity
-inf	minus infinity
NaN	Not-a-Number

1.2.2 Mathematical operators

+	**addition**
-	substraction
*	multiplication
/	division
^	power

```
 >> (2 + 5.2)*10 / (5^3)
ans =
0.5760
>> -2.52e3
ans =
-2520
>> 2*pi
ans =
6.2832
 long format e displays 16 digits:
>> format long e
>> 2*pi
ans =
6.283185307179586e+000
 format short (format par défaut) affiche 5 chiffres :
>> format short
>> 2*pi
ans =
6.2832
>> 1 / 0
Warning: Divide by zero
ans =
Inf
>> -1 / 0
Warning: Divide by zero
```

```
ans =
-Inf
>> 0 / 0
Warning: Divide by zero
ans =
NaN
```

1.2.3 Math functions

sin(X)	**sinus**
asin(X)	sinus reverse
cos(X)	cosinus
acos(X)	cosinus reverse
tan(X)	tangent
atan(X)	tangent reverse

exp(X)	**exponential**
log(X)	Natural logarithm
log10(X)	decimal logarithm
sqrt(X)	square root
abs(X)	absolute value

```
 >> sin(2)
ans =
0.9093
 sinus (45°) :
>> sin(45*pi/180)
ans =
0.7071
>> 1 + exp(2.5)
ans =
13.1825
Utilisation de variables
>> 5*3
ans =
15
>> ans+4
ans =
19
>> a= 2 + log(15)
a =
4.7081
>> b = - 45
b =
-45
>> a * b
ans =
-211.8623
>> c = a - sqrt(abs(b))
c =
-2.0002
```

1.2.4 Calculation on complex numbers

In MATLAB, i and j represent the basic imaginary unit. You can use them to create complex numbers such as 6i+9. You can also determine the real and imaginary parts of complex numbers and compute other common values such as phase and angle [2].

- **Functions:**

i	pure imagination
j	pure imagination
conj(X)	conjugate of the complex number X
real(X)	real part
imag(X)	imaginary part
abs(X)	module
angle(X)	argument (in radians)

Example

```
>> (4 - 2.5i)*(-2 + i)/(1 + i)
ans =
1.7500 + 7.2500i
>> a = 1 + i
a =
1.0000 + 1.0000i
>> b = -2 + 3.5j
b =
-2.0000 + 3.5000i
>> a + b
ans =
-1.0000 + 4.5000i
>> a * b
ans =
-5.5000 + 1.5000i
>> a / b
ans =
0.0923 - 0.3385i
>> conj(a)
ans =
1.0000 - 1.0000i
>> a * conj(a)
ans =
2
>> real(a)
ans =
1
>> imag(conj(a))
ans =
-1
>> abs(a)
ans =
1.4142
>> angle(a)
ans =
0.7854
 sqrt : fonction racine carrée
>> c = 2 - sqrt(3)*i
c =
2.0000 - 1.7321i
>> abs(c)
ans =
2.6458
>> angle(c)
ans =
-0.7137
 Argument en degrés :
>> angle(c)*180/pi
ans =
-40.8934
```

1.2.5 The Problem with clc; clear; close all

clc: cleans up the command window and now one can work without getting confused with the commands for previous runs

First part: Basic elements

clear: erases the variables from previous runs this will reduce chances of error in subsequent runs and the programmer does not have to worry about unnecessary trash variables.

close all: closes all currently open figures. This can be very helpful during subsequent runs of the same script. If the figure from the previous run has not been closed then the subsequent run will plot the data on the already open figure. Which of course is a total waste.

Example

```
clc; close all; clear all; %clears command window
 A = []; %variable A matrix
 for i=1:6
 for j=1:5
 A(i,j) = (i - j) / j^2;
end
 end
A=A'
A =

         0    1.0000    2.0000    3.0000    4.0000    5.0000
   -0.2500         0    0.2500    0.5000    0.7500    1.0000
   -0.2222   -0.1111         0    0.1111    0.2222    0.3333
   -0.1875   -0.1250   -0.0625         0    0.0625    0.1250
   -0.1600   -0.1200   -0.0800   -0.0400         0    0.0400
```