Introduction to MATLAB

I Command and Edit Window

Matlab is an efficient tool used for numerical simuations and to visualize graphics. When opening MATLAB, the window is illustrated in Figure 1 :

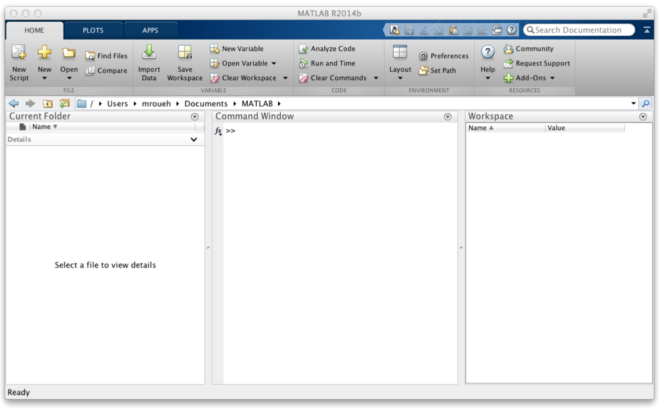

FIGURE 1 – Commnand Window

The commands can be executed in the Command Window. Each command should be followed by "Enter" to be executed. It is recommended to create a script by clicking on "new script". The window in Figure 2 appears and you can create a file containing all the commands. This file should be saved with .m extension and is known as script. The execution of this script can be done either in the command window by typing the name of the file, or by clicking on the button run illustrated in Figure 2.

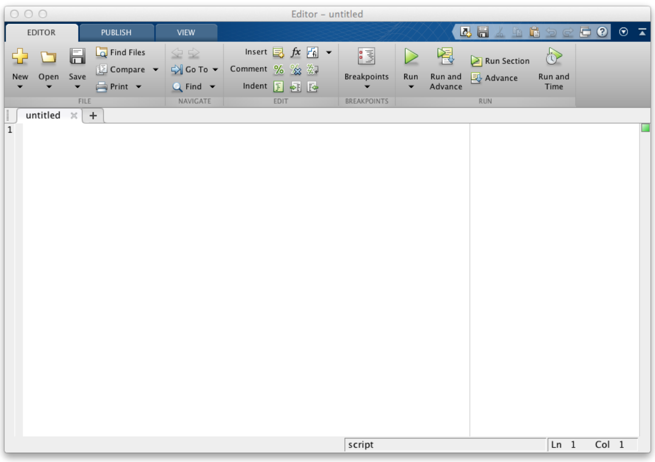

FIGURE 2 – Edit window

II Variables and Operators

In this part, we will use the command window to see how to define variables in MATLAB. After each command, you need to press the button enter on your keyboard. The answer is not displayed when the command is followed by ";" and will be displayed otherwise. The value of the variable is accessible in the workspace by clicking on the name of each variable.

A Scalar Variables and Operation

```
>> a = 5;
>> b = 10;
>> s = a +b;
>> q = a/b;
>> d = a - b;
>> p = a*b;
>> c = a^2
>> r = a^(2/3)
>> sq = sqrt(a)
```

To erase all these variables from the memory, the command is :

```
>> clear all;
```

To erase the displayed values, use

```
>> clc;
```

B Complex numbers

Matlab handles also complex variables. Type the following commands in your Command Window

```
>> a = 1 + 1i;
>> b = 2 + 3i;
>> s = a +b
>> q = a/b
>> d = a - b
>> p = a*b
>> theta = angle(a)
>> rho = abs(a)
>> Re = real(a)
>> Im = imag(a)
>> bara = conj(a)
```

The variables rho et theta contain the polar coordinate of a. Re and Im contain the real and imaginary part of a. bara contains the conjugate value of a.

C Vectors and operations

MATLAB is used also for vector structures.

```
>> V = zeros(1,4);
>> V(1) = 1;
>> V(2) = 2;
>> V(3) = 3;
>> V(4) = 4;
```

To initialize a vector containing n cases, we use the command zeros(1,n) : the table contains one row and n columns. To access to the case k of the table, we use V(k).

```
>> V(3)
```

An other alternative is to fill the table in one shot :

```
>> V = [1,2,3,4];
```

We can also define a mixed vector containing real and complex values :

```
>> V = [1+1i,2+3i,3,4];
```

The transposition of a vector consists to transform a row vector into a column vector. Matlab uses the function transpose :

```
>> transpV = transpose(V)
```

The vectorial operations (sum, difference, scalar product) are as following :

```
>> Vect1 = [1, 2, 3, 4];
>> Vect2 = [5, 6, 7, 8];
>> S = Vect1 + Vect2
>> D = Vect1 -Vect2
>> P = Vect1*transpose(Vect2)
```

An error will be displayed if you forget to make transpose for vect2.
The pairwise product is a MATLAB operation to perform element by element multiplication. Type the following command :

```
>> PP = Vect1.*Vect2
```

The first element of the vector PP is the product of the first element of Vect1 multiplied by the first element of Vect2 and so on...
To fill a vector by regular elements multiple of a given step, we use the following structure :

```
>> a = 1;
>> b = 15;
>> pas = 2;
>> V = a:pas:b
```

If the step is not mentionned, V = a :b, means that the step is 1.

The command length find the length of a vector. The functions max et min find respectively the max and the min values as well as their positions in the vector. The function find gives the position of a given value in a vector V.

```
>> l = length(V)
>> [maxV, indmax] = max(V)
>> [minV, indmin] = min(V)
>> ind = find(V == 3)
```

D Two dimensional table (matrix)

We can also define tables with two dimensions or matrix in Matlab. This table contains con rows and m columns. Suppose we want to define the following matrix in MATLAB

$$A = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{bmatrix}$$

We need to execute this code :

```
>> n = 3;
>> m = 4;
>> A = zeros(n,m);
>> A = [1,2,3,4; 5,6,7,8; 9, 10, 11, 12]
```

The elements are separated by a comma or by a simple space. To disable the display, we need to put ; at the end of the command.
To display the value contained in the second row and the third column, type

```
>> A(2,3)
```

To display all the elements of row 2, type

```
>> A(2,:)
```

The first agrument is a pointer to the row and : indicates to display all the columns.
To display all the elements of column 3, type :

```
>> A(:,3)
```

The reserved word "end" represents implicitly the last index :

```
>> A(:,2:end)
```

We can make extraction of matrix using " :"

```
>> A(2:end,1:2:end)
```

We can compute the transpose of a matrix A using the command transpose(A). We can also compute the determinant of the matrix if its square as well as its trace. Finally, we can find its inverse if its determinant is non zero.

```
>> n = 2;
>> m = 2;
>> A = zeros(n,m);
>> A = [4,1; -4, 1]
>> det(A)
>> trace(A)
>> inv(A)
>> transpose(A)
```

We can add and multiply matrix if the dimensions are respected :

```
>> A = [4,1; -4, 1]
>> B = [1,2; 2, 1]
>> A + B
>> A*B
```

E Boolean variable

A boolean variable is a binary variable that takes two logical values 0 and 1. To check if a variable is equal to another one, we use "==". To check the non equality, we use "$\sim =$".

```
>> x = 2;
>> cond1 = (x > 1)
>> cond2 = (x == 0)
>> cond3 = (x ~= 0)
>> cond4 = (x < 5)
```

The operator "&&" refers to the logical "and" (example : $1 < x < 5$).
The operator "||" refers to the logical "or" (example : $x > 1$ or $x < 5$ i.e. $x \in \mathbb{R}$).

```
>> cond5 = cond1 && cond4
>> cond6 = cond1 ||  cond4
```

III Loops and condition

A Loop

A.1 Loop "while"

In MATLAB, the loop "while" can be written as following :

```
while (condition)
...
...
end
```

The code will be executed as long as the condition is true. Example :

```
>> clear all
>> clc
>> x=1;
>> while x<10
>>  x = x+1;
>> end
>> x
```

A.2 Loop "for"

The loop "for" structure is

```
for compteur = debut:pas:fin
...
...
end
```

There is no need to increment the loop counter, this is done automatically in MATLAB. After a first round in the loop, the counter is equal to debut + pas. As long as, compteur < fin, the loop is executed.

Example :

```
>> clear all
>> clc
>> for x=1:2:9
>>  y=x
>> end
```

B Condition

B.1 "if" et "if-else"

"if" et "if – else" are used to impose conditions. If the condition is not respected, then the code will not be executed.

```
if "expression" ---
---
elseif "expression"
---
---
else
---
---
end
```

Example :

```
>>clear all
>> low = 0;
>> high = 0;
>> for x=1:10
>> if (x<5)
>>  low=low+1;
>> else
>>   high = high + 1;
>> end
>> end
>> low
>> high
```

B.2 instruction "switch"

In Matlab, the structure of "switch" is a following :

```
 SWITCH expr
 CASE case0,
statement, ... , statement
CASE {case1, case2, case3,...}
statement, ... , statement ...
OTHERWISE, statement, ..., statement
END
```

This instruction is used when expr takes different values among case0, case1, ...

IV Figures and graphics

One of the principle advantages of using MATLAB is the possibility to display the variation of signals.
Exemple :

```
>> x = 0:0.01:2*pi;
>> y = cos(x);
>> figure
>> plot(x,y)
>> xlabel('temps')
>> ylabel('signal')
>> title('cosinus')
```

The displayed window is :

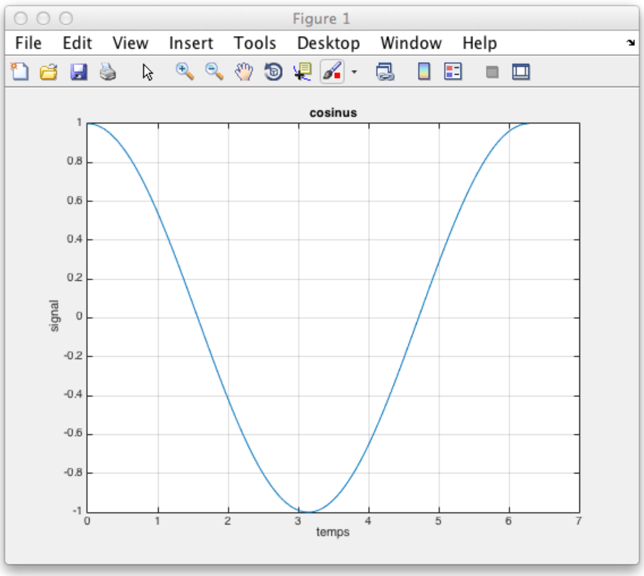

FIGURE 3 – cosinus

To include the graphics in your document, avoid making a screen copy. Instead, do File/save as and then save the figure with the extension .png.
To add other functions to your plot, use the command hold on. To differenciate the curves you can use different colors.

```
>> figure(1)
>> hold on
>> z = sin(x)
>> plot(x,z , '.-r')
>> legend('cosinus', 'sinus')
>> title('cosinus et sinus')
```

The figure is :

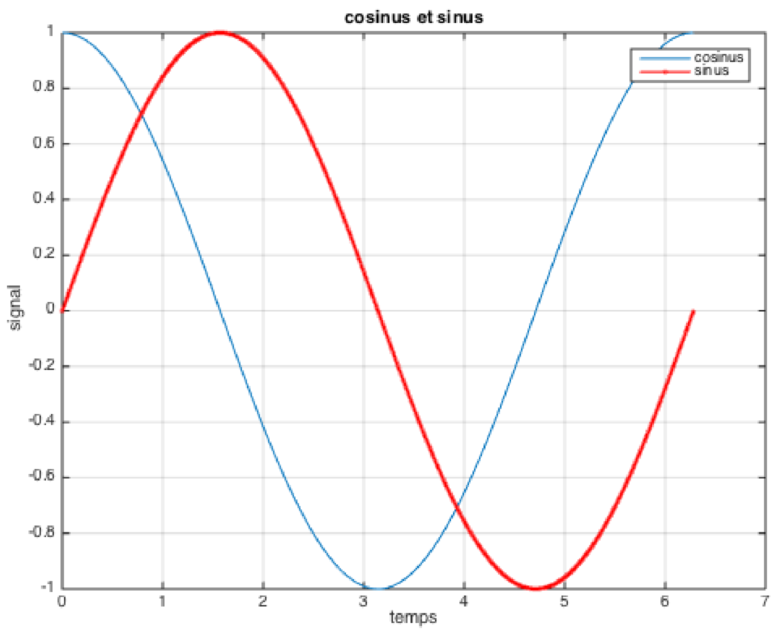

FIGURE 4 – cosinus et sinus : figure with save as + extension .png

In order to not overload MATLAB, it is recommended to close the figures after saving them.

```
>> close all
```

We can finally display two figures on two different graphics, in the same window using subplot :

```
>> x = 0:0.01:2*pi;
>> y = cos(x);
>> z = sin(x);
>> figure
>> subplot(2,1,1)
>> plot(x,y,'b')
>> grid on
>> xlabel('temps')
>> ylabel('signal')
>> title('cosinus')
>> subplot(2,1,2)
>> plot(x,z,'r')
>> grid on
>> xlabel('temps')
>> ylabel('signal')
>> title('sinus')
```

The figure is then :

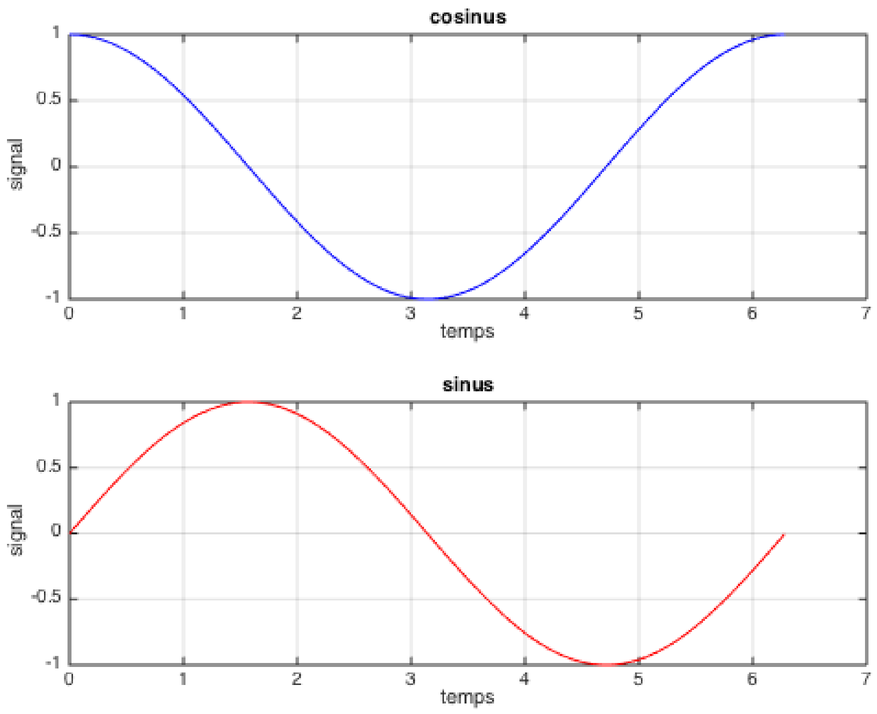

FIGURE 5 – cosinus et sinus

You can always type help in the Command Window if you need details about any command :

```
>> help plot
```