Introduction to MATLAB

MATLAB="MATrix LABoratory"

https://www.mathworks.com/help/matlab/getting-started-with-matlab.html

An alternative to Matlab is the free software Octave, which can be used online

https://octave-online.net

or can be downloaded

https://www.gnu.org/software/octave/download.html

If you download Octave, will be necessary to download and install also the "statistics" package

https://octave.sourceforge.io/statistics/index.html.

1) Basic notions

In Command Window we can perform simple operations or run functions:

```
>> 5+3
ans = 8
>> 1/3
ans = 0.3333
>> sqrt(5)
ans = 2.2361
>> 3^2
ans = 9
```

or we can assign value to variables

```
>> a=10
a = 10
```

We use ; if we do not want to display the result of a command

```
>> b=12;
>>
```

and we use % if we want to write some comments

```
>> 'This line will be displayed' % This will not
ans = This line will be displayed
```

Also we can change the format of the answer:

```
>> format short
>> 1/7
ans = 0.1429
>> format long
>> 1/7
ans = 0.142857142857143
```

Examples of logical operators: ==, $\sim$=, ! =, <, <=, >, >=, &&, ||.

```
>> 4==5
ans = 0
>> 2!=3
ans = 1
```

With $'$disp$'$ we can display a result

```
>> x=2;
>> disp(x/10)
0.2000
```

and with $'$fprintf$'$ we can change the format of the result and display messages

```
>> fprintf('Rezultatului impartirii numarului %d la %d este %4.3f \n', 1, 15, 1/15)
Rezultatului impartirii numarului 1 la 15 este 0.067
```

We use '*help*' or '*doc*' to get informations about functions

```
>> help sqrt
>> doc max
```

2) Matrices

Square brackets [ ] are used to define matrices, $'$;$'$ to delimit rows and $'$,$'$ or $'\sqcup'$ to delimit columns:

```
>> M=[1, 3, 4; 7, 2, 5; 3, 1, 8]
M =

   1   3   4
   7   2   5
   3   1   8
```

In MATLAB we can perform operations with matrices directly:

```
>> A=[1,2; 3,4];
>> B=[5,6; 7,8];
>> A+B
ans =

    6    8
   10   12

>> A-B
ans =

  -4  -4
  -4  -4

>> A+3
ans =

   4   5
   6   7

>> A'
ans =

   1   3
   2   4

>> A*B
ans =

   19   22
   43   50

>> A\B
ans =

  -3.0000  -4.0000
   4.0000   5.0000

>> A/B
ans =

   3.0000  -2.0000
   2.0000  -1.0000
```

The operation $x = A \setminus B$ solve the the linear system $A * x = B$ while $x = A/B$ solve the

the linear system $x * A = B$.

Also it is possible to multiply two matrices A and B by multiplying corresponding elements using $.*$

```
>> A.*B
ans =

    5   12
   21   32
>> A.^2
ans =

    1    4
    9   16
```

There are some "special" matrices in Matlab which can be used for initialization:

```
>> M=zeros(2,3)
M =

   0   0   0
   0   0   0

>> N=ones(3,2)
N =

   1   1
   1   1
   1   1

>> P=eye(3)
P =

Diagonal Matrix

   1   0   0
   0   1   0
   0   0   1

>> Q=magic(4)
Q =

   16    2    3   13
    5   11   10    8
    9    7    6   12
    4   14   15    1
```

For example, *zeros(m,n)* creates a $m \times n$ array of zeros. If we specify only one value as imput, *zeros(n)* will return a square matrix of dimension n.

3) Vectors

In Matlab vectors are also matrices (single row or single column matrices). Hence a vector can be defined as a matrix:

```
>> v=[1, 4, 12, -5, 33]
v =

    1    4   12   -5   33

>> v=[8; 3.6; -17; sqrt(5)]
v =

    8.0000
    3.6000
  -17.0000
    2.2361

>> v=[1 2 3 4]'
v =

   1
   2
   3
   4
```

Also we can define vectors using the colon operator ":". In this way we can create sequences of elements of an arithmetic progressions

```
>> v=1:7
v =

   1   2   3   4   5   6   7

>> w=1:2:9
w =

   1   3   5   7   9

>> x=5:-3:-5
x =

   5   2  -1  -4
```

```
>> y=[0:0.5:3]'
y =

         0
    0.5000
    1.0000
    1.5000
    2.0000
    2.5000
    3.0000
```

Vector operations are very similar to matrix operations, so it is very important for their size to match.

The extraction of some elements from a matrix or a vector is done by specifying the positions of the elements:

```
>> A=[1 2 3; 4 5 6; 7 8 9; 10 11 12]
A =

     1     2     3
     4     5     6
     7     8     9
    10    11    12

>> A(2,3)
ans = 6
>> A(1:2,1:2)
ans =

    1    2
    4    5

>> A(:,3)
ans =

     3
     6
     9
    12

>> A(1,:)
ans =

    1    2    3
```

```
>> A(2:4,:)
ans =

    4    5    6
    7    8    9
   10   11   12

>> A(:)
ans =

    1
    4
    7
   10
    2
    5
    8
   11
    3
    6
    9
   12

>> v=1:9
v =

   1   2   3   4   5   6   7   8   9

>> v(3)
ans = 3
>> v(1:3)
ans =

   1   2   3

>> v(3:6)
ans =

   3   4   5   6

>> poz=[1, 4, 7, 8];
>> v(poz)
ans =

   1   4   7   8
```

We can also concatenate matrices or vectors whose dimensions match:

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

   0   0   0
   0   0   0

>> B=ones(3,3)
B =

   1   1   1
   1   1   1
   1   1   1

>> M=[A;B]
M =

   0   0   0
   0   0   0
   1   1   1
   1   1   1
   1   1   1

>> C=eye(2)
C =

Diagonal Matrix

   1   0
   0   1

>> N=[A,C]
N =

   0   0   0   1   0
   0   0   0   0   1

>> v=1:4
v =

   1   2   3   4

>> w=9:-1:5
w =

   9   8   7   6   5
```

```
>> z=[w,v]
z =

   9   8   7   6   5   1   2   3   4
```

4) M-files: scripts and functions

Everything we run in Command Window will be deleted when we close the work session. If we want to save our work we have the possibility to create Matlab-files (files that contains Matlab code have the extension .m). There are two types of Matlab files: scripts and functions. Both scripts and functions allow us to reuse sequences of commands by storing them in code files. Scripts are the simplest type of code file, since they store commands exactly as you would type them at the command line. However, functions are more flexible and more easily extensible.

Listing 1: script.m

```
% Script example: Computing the area of a circle
R=3; %R=radius of the circle
Area=pi*R^2;
fprintf('The area of the circle with radius %3.2f is %4.3f\n'
   ,R, Area)
```

```
>> script

The area of the circle with radius 3.00 is 28.274
>>
```

We run the scripts when we click on the Run button or by pressing the F5 key.

Listing 2: Function: CircArea.m

```
function S=CircArea(R)
  S=pi*R^2;
endfunction
```

```
>> CircArea(5)
ans = 78.540
>>
```

It is VERY IMPORTANT that function Matlab files to be saved with the same name as the function name.

5) Graphics in MATLAB

The function *plot(X,Y)* creates a 2-D line graph of the data in Y versus the corresponding values in X. IMPORTANT: X and Y has to be vectors of the same length.

Listing 3: Plot 1

```
x = -2 : 0.01 : 2;
y = x.^2;
plot(x,y)
```

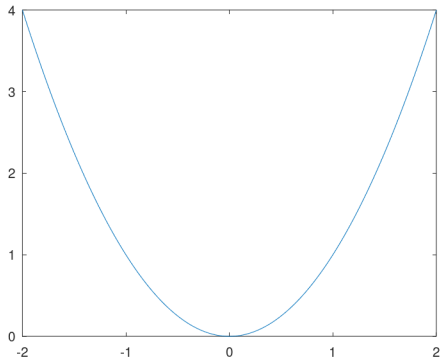

Figure 1: Plot 1

With the command *hold on* we can display two or more graphs in the same figure.

Listing 4: Plot 2

```
x = 0 : pi/100 : 2*pi;
s = sin(x);
c = cos(x);
plot(x,s)
hold on
plot(x,c)
```

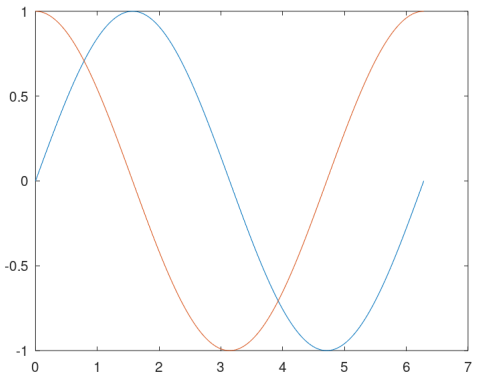

Figure 2: Plot 2

Also, if we want to display more graphs in the same figure but in different windows we can use *subplot* function.

Listing 5: Plot 3

```
x = -1 : 0.01 :1;
f1= x;
f2= x.^2;
f3= x.^3;
f4= x.^4;
subplot(2,2,1)
plot(x,f1)
subplot(2,2,2)
plot(x,f2)
subplot(2,2,3)
plot(x,f3)
subplot(2,2,4)
plot(x,f4)
```

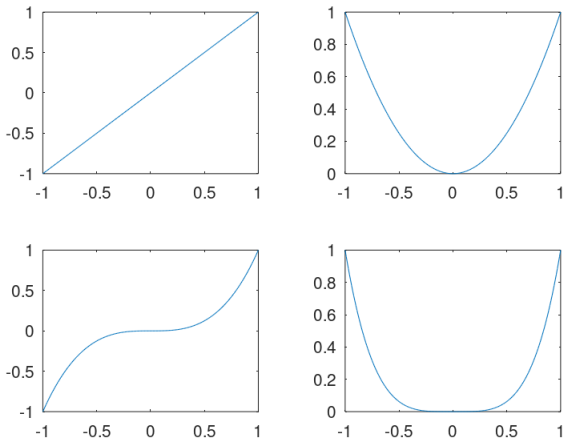

Figure 3: Plot 3

We can specify the line style, line width, color, marker, marker size and we can also add title, axis labels and legend to the graph

Listing 6: Plot 4

```
x = linspace(-1,1,100);
f1= x;
f2= x.^2;
f3= x.^3;
f4= x.^4;
plot(x,f1,'-k','LineWidth',2)
hold on
plot(x,f2,'--b','LineWidth',3)
plot(x,f3,':r','LineWidth',4)
```

```
plot(x,f4,'-o g')
axis equal
title('Functions plot')
xlabel('x')
ylabel('f(x)')
legend('f1','f2','f3','f4','Location','southeast')
```

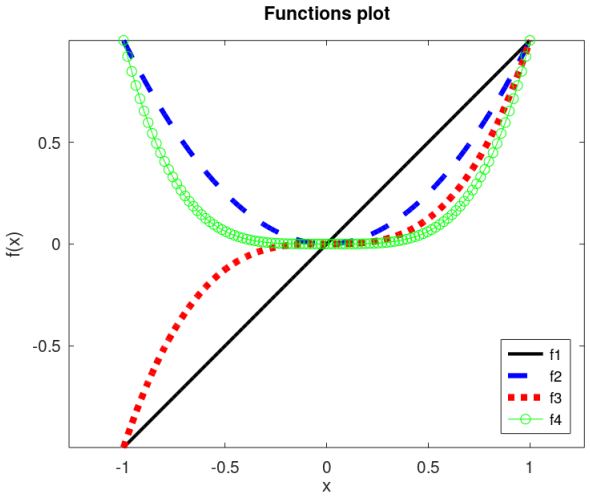

Figure 4: Plot 4

To clear the Command Window we use *clc*, to delete the previous graphics we use *clc*. To delete the values assigned to the variables we can use the *clear all* command.