# Practical work N°1

**Objective:**
Understand the basic commands in Octave.

**Preamble:** Octave **1.0** (1993) → Octave **8.4.0** (November 2023)
GNU Octave is a numerical calculation environment similar to Matlab (very similar syntax), particularly efficient for matrix calculation. It is used in many fields such as data analysis, image processing, automation, electronics, etc.

## 1. Basic commands

- **Help** command name : view the help
- **Edit** command name : view source code
- Comma, semicolon: instruction separator
- Strings between **' '**
- **Name=value** : Assignment
- **X=input**('Give please a value') : read a value % Octave is case-sensitive
- **rand** : random number [0,1]
- **rand(N)** : NxN matrix containing random numbers [0,1]
- To continue an expression on the next line, end the current line with "…"
- Constants: **pi** =3.1416, e = 2.7183, **Inf** = infinity , **eps** = epsilon ($2^{-52}$)…..
- Display formats
  - **format short** displays numbers with 4 digits after the decimal point (default)
  - **format long** displays numbers with 14 digits after the decimal point
  - **format bank** displays numbers with 2 digits after the decimal point
  - **format rat** displays numbers in the form of a ration (a/b)
- Most used functions

**sin(x)** the sine of x (in radians), **cos(x)** the cosine of x (in radians), **sqrt(x)** the square root of x, **abs(x)** the absolute value of x, **round(x)** round a number to the nearest integer, **floor(x)** round a number to the smallest integer, **ceil(x)** round a number to the greatest integer….

## 2. Initialization functions

- **A = [ 4 -2 1 ]** % Creation of a line vector A
- **A = [ 4 ; -2 ; 1 ]** % Creation of a column vector A
- initialization line by line, example: A=[ 1 2 3 ; 4 5 6]
- **A=2 : 15** % consecutive values step 1
- **A=2 :3 :15** % consecutive values step 3
- **A = 0:0.2:1** % consecutive values step 0.2
- **zeros(m,n)**
- **ones(m,n)**
- **eye(m,n)** : ones on the diagonal, zeros elsewhere.

## 3. Size

- **[m n]=size(A)**: number of rows and columns of A
- **m = size (A, 1)** % m contains the number of rows
- **n = size (A, 2)** % n contains the number of columns

## 4. Concatenation

- **C=[A,B]** : concatenate A(m,n) and B(m,p)
- **C=[A;B]** : concatenate A(m,n) and B(p,n)

## 5. Access to the elements of a matrix

- Parentheses ( ) are used for access.

- Square brackets [ ] are used only during creation
- **A(i,j)** : element row i, column j (numbering starts from 1)
- **A(i, :)** : view line i
- **A( :, j)** : view column j
- **A(i1 :i2,j1 :j2)**: submatrix
- **A(2:4,2:end)**
- **A(end,3)** : element A(m,3)
- **A([1 3 7])** % the 1st, 3rd and 7th position
- **A(6) = -3** % assign the value -3 to the sixth element
- **A(2)=[]** % remove the second element
- **A(3:5) = []** % delete from the 3rd to the 5th element
- **A(2,:)** = [] % delete the second line
- **A = [A ; [1 2 -5 9]]** % add new line [1 2 -5 9] to A
-

## 6. Operations

- **C=A+B** : addition
- **C=A-B**: subtraction
- **C=A.*B** : element by element multiplication
- **C=A./B** : element by element division
- **C=A*B** : matrix product
- **C= A+x** : add x for all elements of A
- **C= A*x** : multiply all elements of A by x
- **C= A.^x** : all elements of A to the power of x
- **C = f (A)** : apply the function f on all elements of A
- **A'** : transpose of a matrix A
- **[x y]=min(A)** : x is a vector containing the minimum values of each column
  y: the indices of each element of x
- **[x y]=max(A)**
- **min(min(A))** : the min of the entire matrix
- **max(max(A))** : the max of the entire matrix
- **mean(A)** : the average of each column of A
- **sum(A)** : vector containing the sum of each column
- **prod(A)** : vector containing the product of each column
- **median(A)** : vector containing the median of each column
- **find(condition)** : the indices of the values satisfying the condition
- **a(find(a >0.7)) = 1;** all elements > 0.7 become 1
- **sort(A)** : sorts the elements of a vector or each column of a matrix % see **sort(A,'ascend')** and **sort(A, 'descend')**
- **std2(A)** : the standard deviation of A
- **var(X,1) :** variance of vector X
- **isequal :** tests if two (or more) matrices are equal (having the same elements). It returns 1 if this is the case, and 0 otherwise. Example : **isequal(A,B)**
- **diag (A)** : diagonal of A
- **B=A( :)** : transform A into a vector

## 7. Other commands

- **clear** : clear all variables
- **clc** : clear the command window
- **who** : view all variables
- **whos** : view all variables with details
- **%** : comment

- **tic ……..toc** : execution time in seconds
- **X = repmat(A,L,C)** : replicate the matrix A LxC times
- **reshape :** resizing of a matrix, which consists of modifying the number of rows and/or columns of a matrix while keeping the same number of elements.

example : **Reshape(a,2,3)**

| | | | | | |
|---|---|---|---|---|---|
| 0.7373 | 0.8939 | | 0.7373 | 0.0118 | 0.1991 |
| 0.1365 | 0.1991 | → | 0.1365 | 0.8939 | 0.2987 |
| 0.0118 | 0.2987 | | | | |

## 8. Predefined types

- int8, uint8, int16, uint16, int32, uint32, int64, uint64, single, double, logical, char

## 9. Conditional statements

- **If** condition [...] **end**
- **If** condition [...] **else** [...] **end**
- **if** condition1 [...] **if** condition2 [...] **else** [...] **end else** [...] **end** [...]**end**
- **if** condition1 [...] **elseif** condition2 [...] **elseif** [...] **else** [...]**end**
- Relational operators: **<, <=, >, >=, ==, ~=**
- Logical operators **: &, | , ~ and( , ) or( , ) not( )**
- **switch** …….**end**

## 10. Iteration statements

- **For** i=begin : step : final value ……. **end**
- **while** (expression)…….. **end**

## 11. Functions

- **function [a b c…]=FFF(x,y,z,…)**: create **FFF** function with **x, y, z**,.. as input and **a, b, c**… as output.
- The file name must be the same name of the function with the extension **.m**

## 12. Graphics and data visualization

- The plot function can be used with vectors or matrices.
- **plot(x,y)** : draw a graph (see the options: color, line, etc.)
- format **plot(x, <linestyle><marker><color>)**
  example: **plot(A,'-.*r')**

- linestyle
  '-' Use solid lines (default). '--' Use dashed lines.
  ':' Use dotted lines. '-.' Use dash-dotted lines

- marker
  '+' crosshair 'o' circle '*' star '.' point 'x' cross
  's' square 'd' diamond '^' upward-facing triangle
  'v' downward-facing triangle '>' right-facing triangle
  '<' left-facing triangle 'p' pentagram 'h' hexagram

- **title**('Title of the figure')
- **ylabel**('Y axis')
- **xlabel**('X axis")
- add/delete a grid **grid , grid off**
- **hold on** : draw multiple curves in the same figure.

**Exercise 1**

In Octave, as in most numerical calculation software, by manipulating matrices, we try as much as possible not to use loops to obtain faster programs.

Consider a matrix **M(500x500)** which contains random numbers in the interval **[0, 1].**

1. Write a script that sets to zero all elements of **M** greater than 0.5.
2. Propose two solutions, one with the use of loops and the other with the **find** command.
3. Compare the execution times of the two solutions.

**Exercise 2**

Write an Octave script which allows to:

1. Create a matrix **M(8x4)** containing random integer values in the interval [+25, 100].
2. Show the type of matrix M.
3. Convert M to integer type.
4. Display the submatrix ($2^{nd}$ line: $4^{th}$ and $3^{rd}$: the last column).
5. Calculate the sum of the $3^{rd}$ column.
6. Insert a random line above **M**.
7. Insert a random column to the right of M.
8. Show the size of M.
9. Delete $2^{nd}$ line.
10. Delete a line **L**, where the value of **L** is entered on the keyboard.
11. Delete the last 2 columns.
12. Make all elements of the submatrix null ($2^{nd}$ line: $4^{th}$ and $3^{rd}$: the last column).
13. Assign a value **V** (keyboard entry) to all elements in the last line.
14. Create a function **Modify(M, val)**, which assigns the value **val** to all elements of M in [50, 80].
17. Call the function **Modify (M, val)**, val: keyboard entry.
19. Resize the matrix **M(16x2)**.

**Exercise 3**

1. Write an Octave script that draw in the same graph, the curves of F(x) and G(x) in the interval $[-3\pi, 3\pi]$ with step = 0.05.
2. Add the different notations on the graph (axes, title).
3. Change the different properties of F(x) curve (color line, marker, size...).
4. Draw the 2 curves separately on 2 different figures.