2. *Part Two: The Matlab Programming Language and Using Scripts*

Part Two: The Matlab Programming Language and Using Scripts

2.1 Script files

A script file is used to group series of Matlab commands. This avoids having to enter several times long sequences of instructions. When launching, the instructions it contains sequentially executed as if they were launched from the command prompt. A script stores its variables in the workspace, which is shared by all the scripts. Thus, all the variables created in the scripts are visible from the Window and Vice Versa command. When Matlab detects an error, the program stops and an error message is displayed on the screen (with the number of the line where the error is detected).

Let us edit our script tp1f.m.

Example 1

```
x = cos(10);
y = sin(10);
results (1) = x + y ;
results (2) = x * y ;
results (3) = sqrt(x^2+y^2);
results
```

Our script can then be executed, either by typing its name (without the extension) at the prompt of order, either by clicking on the editor's "RUN" button (icon with a green triangle).

```
>> tp1f
results=

   -1.3831    0.4565    1.0000
```

After running tp1f, the x and y variables are still available in Matlab memory. To be convinced, let's type

>> who

Your variables are:

ans results x y

>> whos

Name	Size	Bytes	Class	Attributes
ans	1x3	24	double	
results	1x3	24	double	
x	1x1	8	double	
y	1x1	8	double	

Example 2

Let's evoke the Matlab editor with the instruction :

>> edit tp2f

This will open the editor and create the file tp2f.m

Enter the following instructions into the tp2f.m file :

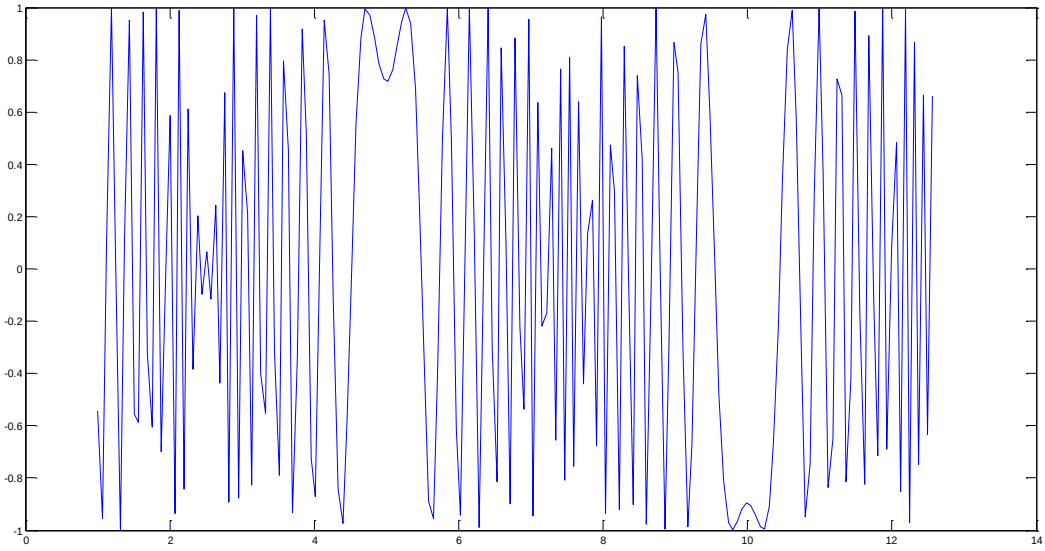
```
clc

x = 1:2*pi/N:4*pi;
```

```
y = sin(w*2*x.^2);
plot(x,y)
```

Next, the command sequence

```
>> N=100;w=5;
>> tp2f
```

produces the figure 1.



2.2 Functions

There is a conceptual difference between functions in computer science and mathematics:

1. In computer science, a function is a routine (a sub -program) which accepts arguments (parameters) and which returns a result
2. In mathematics, a function f is a relation that assigns to each value x at most a single value f(x)

2.2.1. Creating a function in an M-Files

MATLAB contains a big number of prédéfinies functions as sin, cos, sqrt, sum, … etc. and it is possible to create our own functions by writing their source codes in files **M-Files** (carrying the same name of function) by respecting the following syntax:

```
function [y1,...,yN] = myfun(x1,...,xM)
```

function [y1,...,yN] = myfun(x1,...,xM) declares a function named myfun that accepts inputs x1,...,xM and returns outputs y1,...,yN. This declaration statement must be the first executable line of the function. Valid function names begin with an alphabetic character, and can contain letters, numbers, or underscores. You can save your function:

In a function, file which contains only function definitions. The name of the file must match the name of the first function in the file. In a script file which contains commands and function definitions. Functions must be at the end of the file. Script files cannot have the same name as a function in the file. Files can include multiple local functions or nested functions. For readability, use the end keyword to indicate the end of each function in a file. The end keyword is required when: Any function in the file contains a nested function. The function is a local function within a function file,

and any local function in the file uses the end keyword.The function is a local function within a script file.

Example 1

Write a function that calculates the square root of a number by Newton's method

```
function r = tpracine(number)
    r = number/2;
    precision = 6;
    for i = 1:precision
  r = (r + number ./ r) / 2;
    end[80]>> x = tpracine(10)
x =
    3.1623
>> x = tpracine(101)
x =
   10.0499
>> x = tpracine([15 52 166 3])
x =
    3.8730    7.2111   12.8841    1.7321
```

2.3 Structures of control of flux

The flow control structures are instructions to define and manipulate the order of execution of tasks in a program. They offer the possibility of carrying out different treatments depending on the statement of the program data, or making repetitive loops for a given process.

2.3.1 Conditional Statements if, elseif, else

This structure makes it possible to execute an instructions block as a function of the logical value of an expression. Its syntax is:

```
if    expression

 instructions …

end
```

All instructions `instructions` is executed only if `expression` is true. Several exclusive tests can be combined.

```
if expression1

instructions1 …

elseif expression2
```

```
instructions2 …

else

instructions3 …

end
```

Several `elseif` can be concaténés. Their block is carried out if corresponding expression is true and if all previous conditions were not satisfied. The linked block `instruction3` in `else` is as for him carried out if none of the previous conditions was accomplished.

Example 1

We initialize a matrix A as a function of the value of a Nomex variable (example number) as follows:

```
x=input('Enter a value');
y=3;
if x<5
y=0;
elseif x>10
y=2;
end
```

Example 2

```
age = input('Enter your age : ');

   if (age < 3)

  disp('You are a baby')

   elseif (age < 14)

  disp('you are a child')

   elseif (age < 20)

  disp ('You are a teenager')

   elseif (age < 65)

  disp ('you are an adult')

   else

  disp ('You are an oldster')

   endEnter your age : 2
```

```
You are a baby

Enter your age : 10

you are a child

Enter your age : 19

You are a teenager

Enter your age : 59.5

you are an adult

Enter your age : 80

You are an oldster
```

2.3.2 Conditional Statements switch, case

In this structure, a numeric expression is successively compared to different values. As soon as there is identity, the corresponding block of instructions is executed. Its syntax is:

```
switch expression

case valeur1,

 instructions1 …

case valeur2,

 instructions2 …

case valeur3,

 instructions3 …

otherwise

 instructions …

 end
```

The tested expression, an expression, must be a scalar or character string. Once an instructioni block is executed, the execution flow exits the structure and resumes after the end. If no check box checks the tie, the block that follows otherwise is executed.

Example 1

This example does a very simple job. The core idea is to pass through a switch statement and print message based on some condition. We create a basic logic of matching the number and providing an output based on the number.

```
N = input('Enter a number of your choice: ');
switch N
case -2
disp('negative one selected')
case 0
disp('zero selected')
case 2
disp('positive one selected')
otherwise
disp('Some other value')
end
```

Output:At the command prompt, enter the number -2.negative two Repeat the code and enter the number 5. Some other value.

Example 2

In this example of Switch Statement in Matlab, based on the grade obtained, we classify the distinction.

```
Enter_grade = 'A';
switch(enter_grade)
case 'A'
fprintf('Excellent performance!\n' );
case 'B'
fprintf('Well done performance\n' );
case 'C'
fprintf('Very Good performance\n' );
case 'D'
fprintf('You passed.. Congratulations\n' );
case 'F'
fprintf('Better luck next time\n' );
otherwise
fprintf('Invalid grade. Please enter correct value\n' );
end
```

2.3.3 Conditional loop while

This mechanism makes it possible to repeat a series of instructions as long as a condition is verified. Its syntax is:

```
While expression

Instructions
```

```
end
```

Example 1

```
a=1 ;

while (a~=0)

a = input ('Enter a number (0 to finish ');

end
```

This program asks the user to enter a number. If this number is not equal to 0 then the loop repeats, otherwise (if the given value is 0) then the program stops.

Example 2

```
eps = 1;

 while (1+eps) > 1

 eps = eps/2;

end
```

Example 3

Find the first number whose factorial uses more than 100 digits:

```
 n = 1;
while prod(1:n) < 1e100
 n = n + 1;
 end
```

Example 4

2.3.4 Iterative loop for

The for statement repeats the execution of a group of statements a specified number of times.
It has the following general form:

```
for variable = expression

statement

...

statement

end
```

Example 1

```
k = input('Enter a number of your choice: ');

a=zeros(k,k) % Preallocate matrix

for m = 1:k

for n = 1:k

a(m,n) = 1/(m+n -1);

end

end
```

Example 2

```
n = 4

x = []

for i = 1:n

 x = [x, i^2]

end
```

Example 3

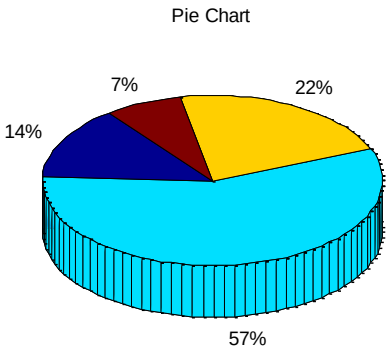
```
clc

m = 4

n = 3

for i = 1:m

 for j = 1:n

H(i,j) = 1/(i+j-1) ;

 end

end
```

```
H

>> m =

 4

n =

3

H =
    1.0000    0.5000    0.3333    0.2500
    0.5000    0.3333    0.2500    0.2000
    0.3333    0.2500    0.2000    0.1667
    0.2500    0.2000    0.1667    0.1429
    0.2000    0.1667    0.1429    0.1250
    0.1667    0.1429    0.1250    0.1111
```

Example 4

Determine the type of plot to create based on the plot type value. If the plot type is "main" or "pie5", create a 3D pie chart. Use an array of cells to hold the two values.

```
clc
x = [15 62 24 8];
plottype = 'pie5';

switch plottype
    case 'bar'
        bar(x)
        title('Bar Graph')
    case {'pie','pie5'}
        pie3(x)
        title('Pie Chart')
    otherwise
        warning('Unexpected plot type. No plot created.')
end
```



2.4 Examples of Applications

2.4.1 Digital integration

Numerical integration is used when an integral is impossible or difficult to solve analytically.

- **Trapeze method**

The approach is to subdivide the interval on which we want to integrate and approximate the integral of the function with a certain number of trapezius. Each subdivision is of equal width h, where h = (b-a)/n where N is the number of subdivisions.

$$\int_a^b f(x) \approx \frac{h}{2}[f(a) + 2f(a+h) + 2f(a+2h) + 2f(a+3h) + \cdots 2f(a+(n-1)h) + f(b)]$$

- **MATLAB Program**

Calculate the integral using the trapezius method $I = \int_1^6 f(x) = e^{-x^2}$.

```
% trapex.m test program for numerical integration using the
composite
% trapezoidal rule to solve the integral of exp(-x^2) between
% a and b
clear; help trapex;
format long; % configures MATLAB to report numbers with more
decimal places
a=input('input a (starting value)->');
b=input('input b (end value) ->');
n=input('input number of intervals (n) ->');
h=(b-a)/n;
fa=exp(-a^2); % f(a)
fb=exp(-b^2); %f(b)
ff=0;
for i=2:n
ff=ff+(2*exp(-(a+(h*(i-1)))^2)); % sum of 2f(a + i(h)) where i = 1
to n-1
end
result=(h/2)*(fa+fb+ff)
% result = f(a) + f(b) + sum of 2f(a + i(h)) where i = 1to n-1
>> input a (starting value)->1
input b (end value) ->6
input number of intervals (n) ->50

result =

   0.140016129321009
```

- **Method of Simpson**

The Simpson integration method is based on the interval division $[x_i, x_{i+1}]$ in three, so obtain an equal step h/3.
The Simpson integration formula can be given by:

$$\int_a^b f(x) = \frac{h}{3}\left[f(x_1) + f(x_{n+1}) + 4\sum_{j=2(j\ paire)}^{n} f(x_i) + 2\sum_{j=3(j\ impaire)}^{n-1} f(x_i)\right]$$

- **MATLAB Program**

Calculate the integral using the simpson method $I = \int_1^6 f(x) = e^{-x^2}$.

```
% Simpsonex.m - A program for composite Simpsons 1/3 rule
% to numerically integrate a function between a and b with
% n subintervals.
% The example program integrates exp(-x^2)
format long % sets MATLAB to report more decimal places
clear; help simpsonex;
a=input('input a (starting value)->');
b=input('input b (end value) ->');
n=input('input number of intervals (n) ->');
h=(b-a)/n; % interval with
fa=exp(-(a^2)); % f(a)
fb=exp(-(b^2)); % f(b)
ff=0;
for i=2:2:n; % all 4*f(a+nh) terms to f(b) h=(1,3,5,7,…..,n-1)
x = (a+(i-1)*h);
fx = exp(-x^2);
ff = ff + 4*fx;
end
for i=3:2:n; % all 2*f(a+nh) terms to f(b) h=(2,3,4,6,…..,n-2)
x = (a+(i-1)*h);
fx = exp(-x^2);
ff = ff + 2*fx;
end
result=(h/3)*(fa+fb+ff) % integral result
% with approximation to area under curve
```

```
>>input a (starting value)->1

input b (end value) ->6
```

```
input number of intervals (n) ->50


result =

   0.139401977198315
```

2.5.2 Laplace transformation

The Laplace transform of a time function f(t) is given by the following integral:

$$I\{f(t)\} = \int_0^\infty f(t).e^{-st}dt$$

The Laplace transform is also called transformed from f (t) into F(s). You can see that this process of transformation or integration converts f (t), a function of the symbolic variable t, in another function F(s), with another variable s.

The transformation of Laplace transforms the differential equations into algebraic equations. To calculate a laplace transform of a function f (t), write:

```
laplace (f(t))
```

Example 1

In this example, we will calculate the Laplace transform of certain commonly used functions. Create a script file and type the following code:

```
clc
syms s t a b w
 laplace(a)
laplace(t)
laplace(t^3)
laplace(exp(b*t))
laplace(sin(2*w*t))
laplace(cos(2*w*t))
ans =
1/s^2
  ans =
 1/s^2
  ans =
 6/s^4
  ans =
 1/(s-b)
 ans =
 2*w/(s^2+4*w^2)
  ans =
 s/(s^2+4*w^2)
```

- **The reverse Laplace transform**

Matlab allows us to calculate the reverse Laplace transform using the `ilaplace` command.

Example 1

```
clc
 syms s t a b w
ilaplace(1/s^4)
ilaplace(3/(w-s))
ilaplace(1/(s^3+4))
ilaplace(exp(-b*t))
ilaplace(2*w/(s^2 - w^2))
ilaplace(2*s/(s^2 + w^2))
ans =
1/6*t^3
 ans =
-3*exp(w*t)
 ans =
1/12*exp(-2^(2/3)*t)*2^(2/3)+1/12*exp(1/2*2^(2/3)*t)*(-
cos(1/2*3^(1/2)*2^(2/3)*t)+3^(1/2)*sin(1/2*3^(1/2)*2^(2/3)*t))*2^(2
/3)
 ans =
ilaplace(exp(-b*t),t,x)
 ans =
2*sinh(w*t)
  ans =
2*cos(w*t)
```

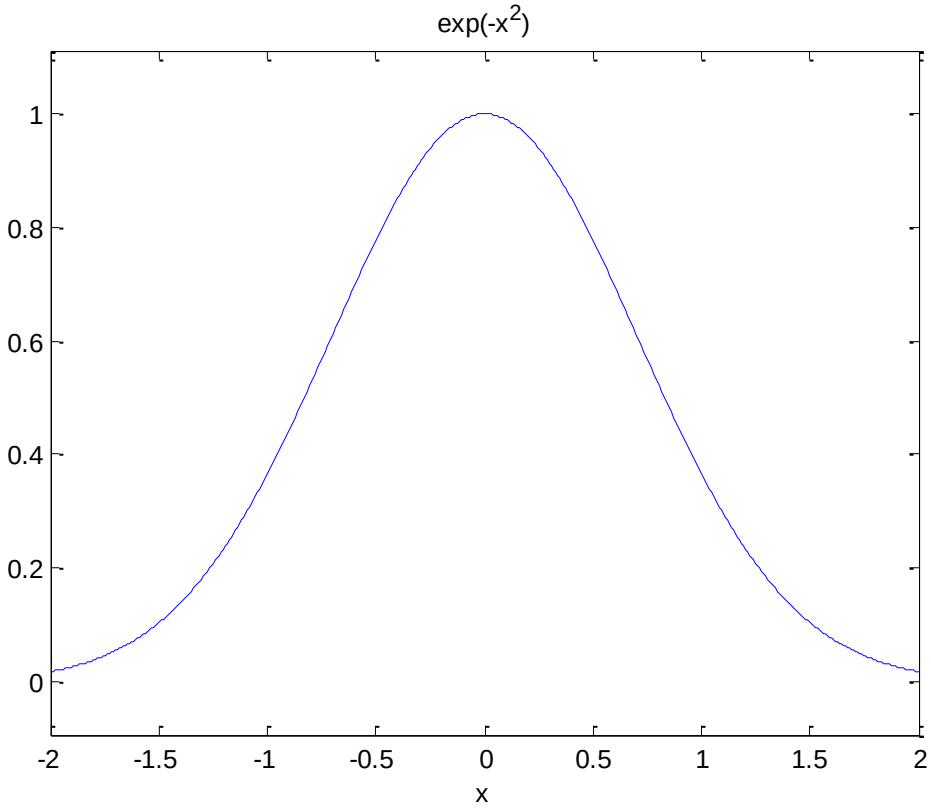
- **The Fourier Transforms**

Fourier's transforms commonly transform a mathematical function of time, f (t), into a new function, sometimes noted or f, whose argument is the frequency with cycles/s (hertz) or radians units per second. The new function is then called Fourier transform and/or frequency spectrum of the function f.

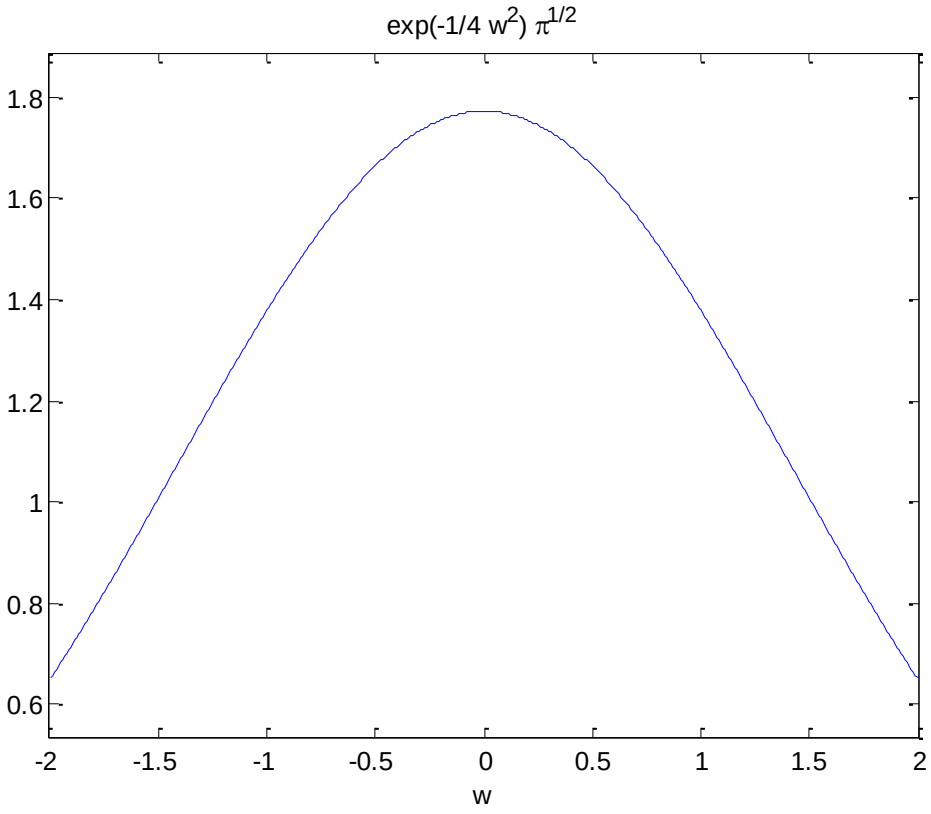
Example 1

Create a script file and type the following code:

```
clc
syms x
f = exp(-x^2);      %our function
ezplot(f,[-2,2])      % plot of our function
FT = fourier(f)       % Fourier transform
```




```
FT =
exp(-1/4*w^2)*pi^(1/2)
Tracé de la transformée de Fourier comme :
ezplot(FT,[-2,2])
```



- **Reverse Fourier transformed**

Matlab provides the Fourier command to calculate the reverse Fourier transform of a function.

Example 1

Find the Inverse Fourier Transform of exp(-w^2/4)

```
% MATLAB code specify the variable
% w and t as symbolic ones
syms w t

% define Frequency domain function X(w)
X=exp(-w^2/4);

% ifourier command to transform into
% time domain function x(t)
% using 1st syntax, where by default
% independent variable = w
% and transformation variable is x .
x1 = ifourier(X);

% using 2nd syntax, where transformation variable = t
x2 = ifourier(X,t);

% using 3rd syntax, where independent variable
```

```
% = w (as there is no other
% variable in function) and transformation
% variable = t
x3 = ifourier(X,w,t);

% Display the output value
disp('1. Inverse Fourier Transform of exp(-w^2/4) using ifourier(X)
:')
disp(x1);

disp('2. Inverse Fourier Transform of exp(-w^2/4) using
ifourier(X,t) :')
disp(x2);

disp('3. Inverse Fourier Transform of exp(-w^2/4) using
ifourier(X,w,t) :')
disp(x3);
x1=exp(-x^2)/pi^(1/2)
x2=exp(-t^2)/pi(1/2)
x3=ex(-t^2)/pi^1/2).
```

Reference

[1] Matlab Guide, D. J. Higham, N. J. Higham, SIAM, 2000.
[2]Introduction to Scientific Computing, A Matrix-Vector Approach Using MATLAB, C.F. Van Loan, MATLAB curriculum Series, 1997.
[3]Apprendre et Maîtriser Matlab, versions 4 et 5 et SIMULINKr, M. Mokhtari, A. Mesbah, Springer, 1997.

[4] The MATHWORKS, Inc., Using MATLAB Graphics, The Mathworks, Inc., Natick, MA, 1996.

Some internet sites

- le site officiel de Matlab https://www.mathworks.com/
- https://www.tutorialspoint.com/matlab/matlab_transforms.htm
- http://engineering.nyu.edu/mechatronics/vkapila/matlabtutor.html
- http://matlab.izmiran.ru/help/toolbox/webserver