Support de Travaux Pratiques Traitement numérique de signal médical

Niveau 3éme année Licence Appliquée en Génie Biomédical

Etablissement ISBS Sfax

Année universitaire 2018-2019

Ministère de l'Enseignement Supérieure et de la Recherche Scientifique
Université de Sfax
Institut Supérieur de Biotechnologie de Sfax

Travaux pratiques

Traitement numérique de signal médical

2018 - 2019

3éme année Licence appliquée en Génie Biomédicale

Enseignants :

Nessrine ELLOUMI, Assistante Vacataire à ISBS de Sfax

Pr. Mohamed Salim BOUHLEL, Professeur à ISBS de Sfax

Sommaire

Ministère de l'Enseignement Supérieure et de la Recherche Scientifique
Université de Sfax
Institut Supérieur de Biotechnologie de Sfax

Traitement numérique de signal médical

TP1 : Initiation à MATLAB

2018 – 2019

3éme année Licence appliquée en Génie Biomédicale

Enseignants :

Nessrine ELLOUMI, Assistante Vacataire à ISBS de Sfax

Pr. Mohamed Salim BOUHLEL, Professeur à ISBS de Sfax

TP1 : Initiation à MATLAB

I. Introduction

- Se familiariser avec le logiciel MATLAB.
- Manipuler les variables dans l'environnement de travail.

II. Présentation de l'interface graphique Matlab

La fenêtre d'accueil de Matlab se compose de trois fenêtres :

La fenêtre ***« Ligne de commande »*** à droite est l'espace des lignes de commandes (). Cet espace permet à l'utilisateur de programmer et d'afficher des résultats en tapant directement les commandes. Chaque commande exécutée sous cette fenêtre sera copiée dans la fenêtre de l'historique des commandes (***Command History***).

La fenêtre « ***Historique des commandes »*** (en bas à gauche) permet de sauvegarder les commandes utilisées précédemment dans la fenêtre de ligne de commandes.

Matlab dispose d'une barre des menus qui permet de manipuler ce logiciel, une barre d'outils qui permet d'accéder aux outils les plus outils utilisés et de basculer d'un mode d'usage de Matlab vers un autre (M-file, Simulink, GUI).

La partie « ***Current Directory »*** affiche le nom du dossier en cours d'utilisation.

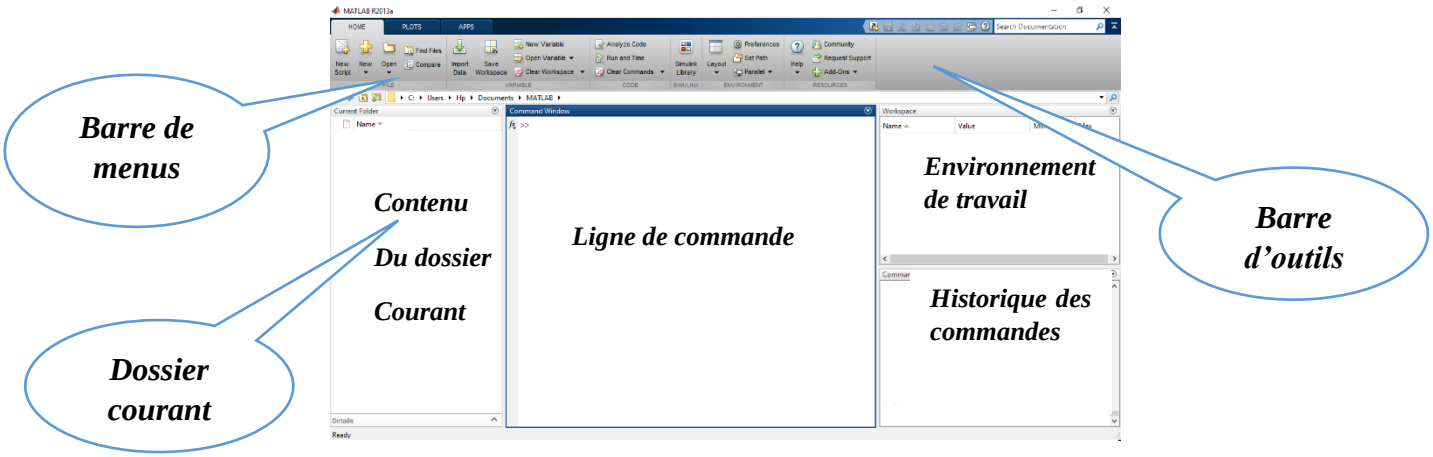


Figure 1. *Présentation de l'accueil de Matlab*

III. Manipulation

1. Aspects élémentaires

1.1. Aides

```
>> help  fonction % affiche l'aide d'une fonction.
>> help desk  % affiche la documentation en hypertexte ;
>> help win  % affiche aide en ligne dans une fenêtre séparée ;
>> lookfor mot clé  % recherche d'un mot clé ;
>> which  % localise fonctions et fichiers ;
>> what  % liste des fichiers Matlab dans le répertoire courant ;
>> exist  fonction  % check si une fonction ou une variable existe dans le Workspace.
```

1.2. Mode Interactif

En mode commande, MATLAB affiche des chevrons >> et attend une commande :

```
>> 1+3
4
>> 3^3
27
>> 2*sin (pi/4)
1.4142
```

Le dernier résultat est stocké dans la variable **ans** qu'il est possible de réutiliser :

```
>> ans^2
2.0000
>> ans-2
-4.4409e-16
```

1.3. Commentaires, ponctuations

```
>> 2+3
5
>> 2*2 ;
4
```

- Le caractère ';' sert à supprimer l'affichage d'une ligne de commande.
- Le caractère '%' sert à afficher des commentaires.

1.4. Variables scalaires, opérations élémentaires

```
>> var=2
2
>> autre=3 ;
>> who
Your variables are:
ans    autre  var
>> whos
Name       Size          Bytes  Class     Attributes
 ans        1x1             8  double
 autre      1x1             8  double
 var        1x1             8  double
>> clear autre
>> who
Your variables are:
ans  var
>> clear
>> clc
>> clear all
```

- ***Opérations élémentaires :*** **+ - * / or \ ^**

```
>> 4/2
2
>> 4\2
2
```

1.5. Les variables

Les résultats peuvent être stockés dans les variables :

```
>> x=pi/3
x =
   1.0472
```

```
>> cos(x)
ans =
   0.5000
>> y=sin(x)^2+cos(x)^2
y =
    1
```

1.7. Entrées-sorties

```
>> N = input ('valeur de N=')
Valeur de N=
>> disp ("ceci est un test") ;
ceci est un test
```

2. Vecteurs et matrices

2.1. Vecteurs

a) Création de vecteurs

Un vecteur ligne se déclare entre crochets en séparant les éléments avec des espaces ou des virgules :

```
>> a= [4 5 3]
a =
    4    5    3
>> E= a(1)
4
```

Pour créer un vecteur colonne, on utilise le séparateur point- virgule :

```
>> z = [4;5;3]
z =
    4
    5
    3
```

Il existe des commandes pour créer des vecteurs d'une manière automatique:

```
>> début=0
>> fin=3
>> pas =0.5
```

```
>> V= début: pas : fin
>> V1= 0 :0.3 :1.1
V1 [ 0 0.3000 0.6000 0.9000].
```

Remarque : Les deux points (:) permettent de générer des vecteurs rempli de valeurs équidistantes.

b) Accès aux valeurs

L'accès aux valeurs des vecteurs (pour les lire ou les écrire) se fait à l'aide des parenthèses pour indiquer l'élément souhaité dans le vecteur. Un indice en dehors d'un tableau entraîne une erreur :

```
>> a(2)
5
>> a(2) = 7
a =
   4    7    3
>> a(6)
Index exceeds the number of array elements (3).
```

c) Transposition

L'opérateur apostrophe est utilisée pour créer un vecteur **colonne,** c'est l'opérateur transposition :

```
>> tab=a'
tab =
   4
   7
   3
>> z = [3 5 6]'
z =
   3
   5
   6
```

d) Combinaison de vecteurs

```
>> b = [1:3]
1 2 3
```

```
>> c= [10 :10 :30]
c =
   10    20    30
>> d= [b c]
d =
    1     2     3    10    20    30
```

d) Opérations scalaires

```
>> a= [4  5  3]
a =
    4     5     3
>> s= sum(a)
s =
   12
>> p= prod(a)
p =
   60
>> m= mean(a)
m =
    4
>> r= sqrt(a)
r =
   2.0000    2.2361    1.7321
>> f= [0 -1 3]
f =
    0    -1     3
>> som= a+f
som =
    4     4     6
>> clear all
>> clc
```

2.2. Matrices

a) Création de matrices

Une matrice est un ensemble de lignes comportant toutes le même nombre de colonnes. Les matrices se déclarent comme les vecteurs, en séparant les colonnes par espace et les lignes par point- virgule :

```
>> x1= [1 2 3 ; 4 5 6 ; 7 8 9]
x1 =
   1   2   3
   4   5   6
   7   8   9
>> x2= [1 :1 :3 ; 4 :1 :6 ; 7 :1 :9]
x2 =
   1   2   3
   4   5   6
   7   8   9
>> A= x1(2,3)
A =
   6
>> A1=x1(:,2)
A1 =
   2
   5
   8
>> A2=x2(2,:)
A2 =
   4   5   6
```

c) Opérations entre matrices

- ***Sommations***

```
>> y1= [0 0 1 ; 3 3 4 ; 1 1 0]
y1 =
    0    0    1
    3    3    4
    1    1    0
>> P= x1+y1
P =
    1    2    4
    7    8   10
    8    9    9
>> R= y1+x1
R =
    1    2    4
    7    8   10
    8    9    9
```

- ***Soustractions***

```
>> Y= y1-x1
Y =
   -1   -2   -2
   -1   -2   -2
   -6   -7   -9
```

- ***Multiplications***

```
>> x1= [1 2 3 ; 4 5 6 ; 7 8 9]
x1 =
    1    2    3
    4    5    6
    7    8    9
>> z1= [1 :1 :3 ; 11 :1 :13]
```

```
z1 =
    1     2     3
   11    12    13
>> x1*z1
```

% cette opération génère une erreur parce que les taille des matrices sont diffèrent.

```
Error using  *
Incorrect dimensions for matrix multiplication. Check that the number of columns in the first matrix matches the number of rows in the second matrix. To perform elementwise multiplication, use '.*'.
>> D= x1*z1'
D =
   14    74
   32   182
   50   290
>> a1=[1 0 -1; 3 2 1; 4 4 4]
a1 =
    1     0    -1
    3     2     1
    4     4     4
>> f1=[0 0 1; 3 3 4; 1 1 0]
f1 =
    0     0     1
    3     3     4
    1     1     0
>> pro1=a1*f1
pro1 =
   -1    -1     1
    7     7    11
   16    16    20
```

```
>> pro2=f1*a1
pro2 =
    4    4    4
   28   22   16
    4    2    0
```

- ***Multiplication élément par élément***

```
>> x1= [1 2 3 ; 4 5 6 ; 7 8 9]
x1 =
    1    2    3
    4    5    6
    7    8    9
>> y1= [0 0 1 ; 3 3 4 ; 1 1 0]
y1 =
    0    0    1
    3    3    4
    1    1    0
>> K= x1.*y1     % il faut que les deux vecteurs x1 et y1 ont les mêmes dimensions.
K =
    0    0    3
   12   15   24
    7    8    0
>> G= x1^2
G =
   30   36   42
   66   81   96
  102  126  150
>> J= x1.^2
J =
    1    4    9
   16   25   36
   49   64   81
```

Remarque : L'opération x1*z1' est la multiplication matricielle des matrices x1 et z1' de dimensions respectives M.N et N.P, tandis que x1.*y1 est la multiplication élément par élément de 2 matrices de dimensions identiques.

- ***Divisions***

```
>> z1= [1 :1 :3 ; 11 :1 :13]
z1 =
    1    2    3
   11   12   13
>> z2= [4 5 6 ; 14 15 16]
z2 =
    4    5    6
   14   15   16
>> z1./z2      % chaque élément de z1 est divisé par l'élément équivalent de z2
ans =
   0.2500   0.4000   0.5000
   0.7857   0.8000   0.8125
>> z1.\z2      % chaque élément de z2 est divisé par l'élément équivalent de z1
ans =
   4.0000   2.5000   2.0000
   1.2727   1.2500   1.2308
```

d) Matrices particulières

```
>> N= ones (3,2)  % ones (n,m) : Matrice de taille n*m ne contenant que des 1.
N =
    1    1
    1    1
    1    1
>> U= zeros(2,5)  % zeros (n,m) : Matrice de taille n*m ne contenant que des 0.
U =
    0    0    0    0    0
    0    0    0    0    0
```

```
>> Q= rand (3,2)   % rand (n,m) : Matrice de taille n, m contenant des nombres aléatoires entre 0 et 1.
Q =
   0.8147    0.9134
   0.9058    0.6324
   0.1270    0.0975
>> D= diag([1 :4])   % diag ([A:B]) : Matrice dont les valeur de leur diagonal sont de A a B.
D =
    1    0    0    0
    0    2    0    0
    0    0    3    0
    0    0    0    4
```

e) Caractéristiques des matrices

```
>> x1= [1 2 3 ; 4 5 6 ; 7 8 9]
x1 =
    1    2    3
    4    5    6
    7    8    9
>> [m,n]= size(x1)
m =
    3
n =
    3
>> length(x1)
ans =
    3
```

f) Manipulation de matrices et sous-matrices

```
>> A1= [1 2 3 ; 4 5 6 ; 7 8 9]
A1 =
   1   2   3
   4   5   6
   7   8   9
>> A2= [1, 2, 3 ; 4, 5, 6 ; 7, 8, 9]
A2 =
   1   2   3
   4   5   6
   7   8   9
>> A2 (3, 1)=21 ; A2
A2 =
   1   2   3
   4   5   6
  21   8   9
>> C1= [A1, A2]
C1 =
   1   2   3   1   2   3
   4   5   6   4   5   6
   7   8   9  21   8   9

>> C2= [A1 ; A2]
C2 =
   1   2   3
   4   5   6
   7   8   9
   1   2   3
   4   5   6
  21   8   9
```

```
>> xy= [1:3, 5:7, 9:11]
xy =
    1    2    3    5    6    7    9   10   11
>> xy= [1:3 ; 5 :7 ; 9 :11]
xy =
    1    2    3
    5    6    7
    9   10   11
```

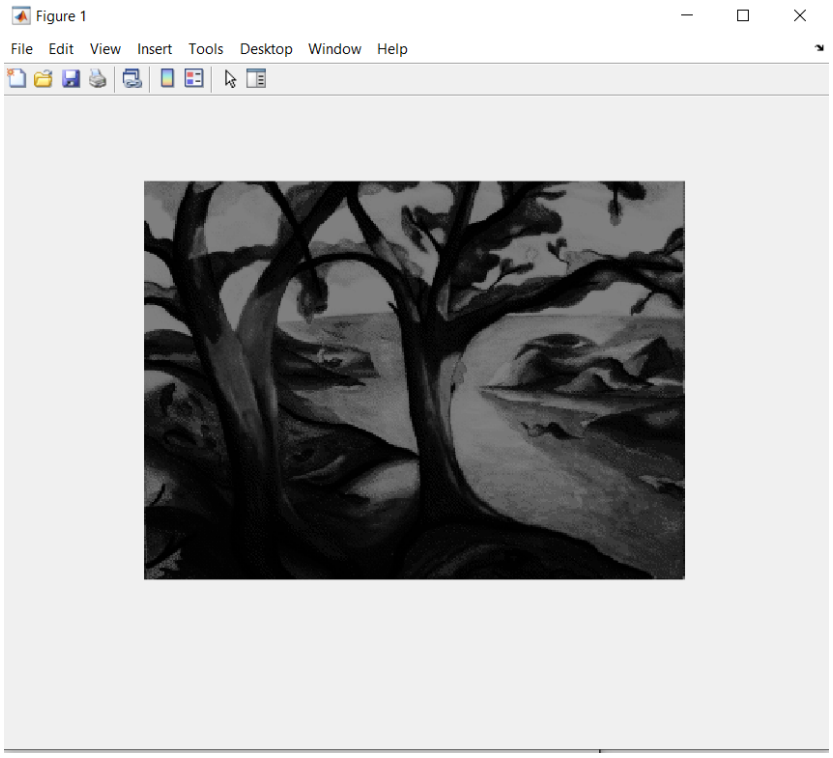
- ***Extraction de vecteurs et sous matrices :***

```
>> xy(  :, 2)
ans =
    2
    6
   10
>> xy ( :, 1:2)
ans =
    1    2
    5    6
    9   10
```

Affichage graphique

Etant donné deux vecteurs de même taille, x et y, la fonction plot(x, y) trace le graphe de y en fonction de x : y= f(x).

```
>> x= 0 :0.1 :2*pi;
x =
  Columns 1 through 11
         0    0.1000    0.2000    0.3000    0.4000    0.5000    0.6000    0.7000    0.8000    0.9000
1.0000
  Columns 12 through 22
    1.1000    1.2000    1.3000    1.4000    1.5000    1.6000    1.7000    1.8000    1.9000    2.0000
2.1000
  Columns 23 through 33
```

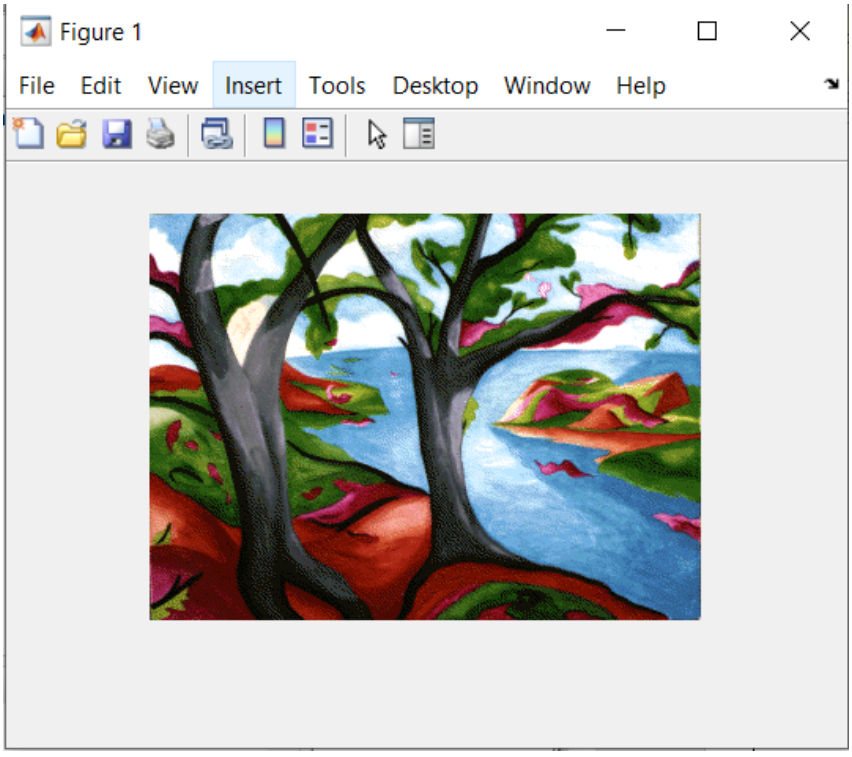


Pourquoi est-elle en niveaux de gris ?

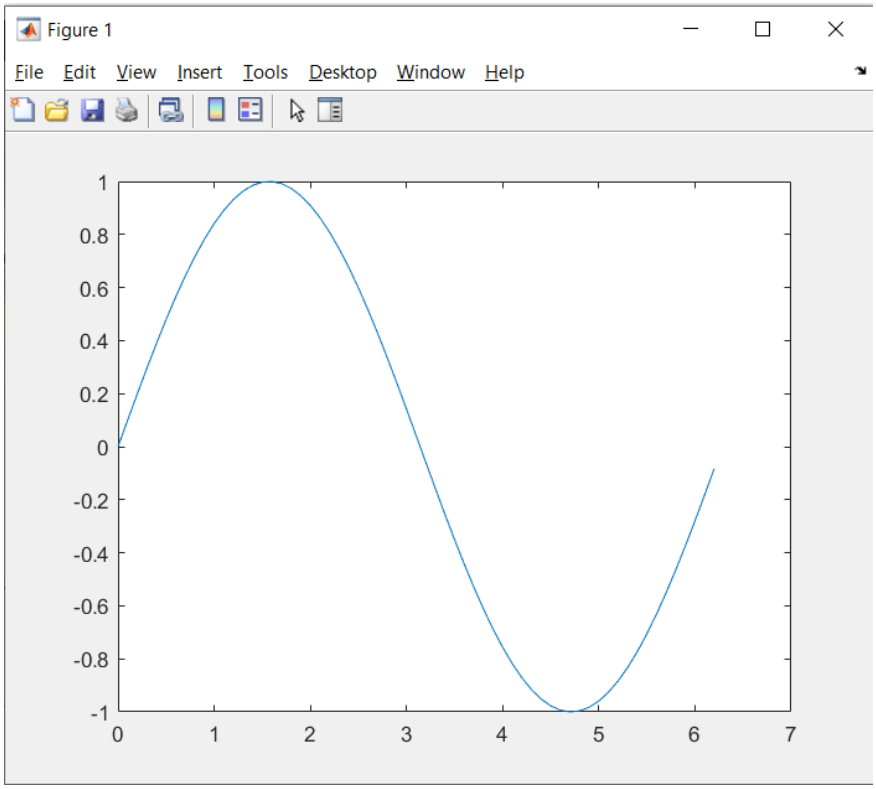
Parce que la variable I contient les niveau de gris de l'image trees

% Afficher l'image trees selon la carte de couleur « LUT »

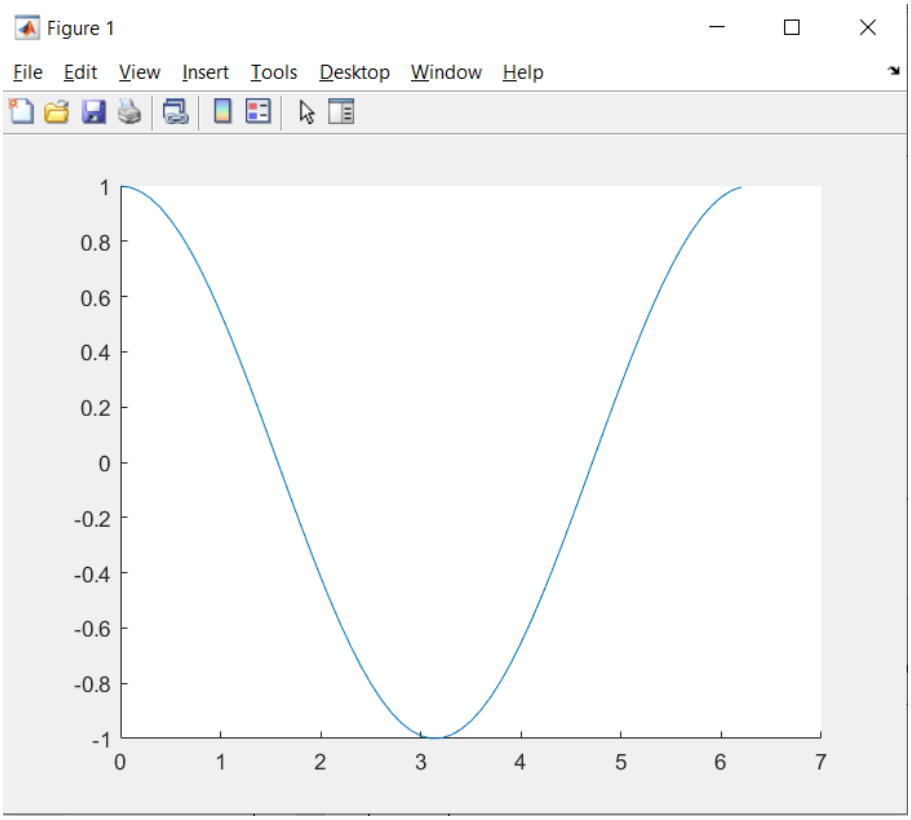
>>imshow(I, LUT)



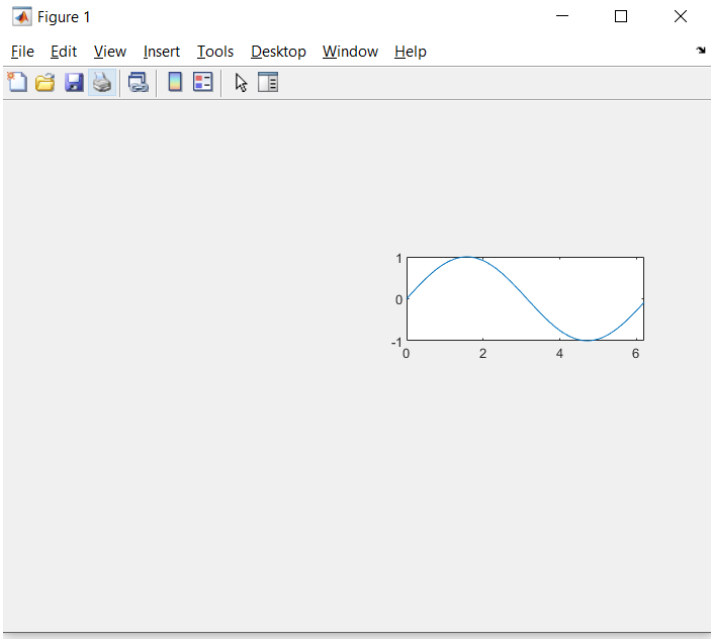
```
   2.2000   2.3000   2.4000   2.5000   2.6000   2.7000   2.8000   2.9000   3.0000   3.1000
3.2000

Columns 34 through 44

   3.3000   3.4000   3.5000   3.6000   3.7000   3.8000   3.9000   4.0000   4.1000   4.2000
4.3000

  Columns 45 through 55

   4.4000   4.5000   4.6000   4.7000   4.8000   4.9000   5.0000   5.1000   5.2000   5.3000
5.4000

 Columns 56 through 63

  5.5000   5.6000   5.7000   5.8000   5.9000   6.0000   6.1000   6.2000

>> y= sin(x) ;

y =

  Columns 1 through 11

         0    0.0998    0.1987    0.2955    0.3894    0.4794    0.5646    0.6442    0.7174    0.7833
0.8415

  Columns 12 through 22

   0.8912    0.9320    0.9636    0.9854    0.9975    0.9996    0.9917    0.9738    0.9463    0.9093
0.8632

  Columns 23 through 33

   0.8085    0.7457    0.6755    0.5985    0.5155    0.4274    0.3350    0.2392    0.1411    0.0416
-0.0584

  Columns 34 through 44

  -0.1577   -0.2555   -0.3508   -0.4425   -0.5298   -0.6119   -0.6878   -0.7568   -0.8183   -0.8716
-0.9162

  Columns 45 through 55

  -0.9516   -0.9775   -0.9937   -0.9999   -0.9962   -0.9825   -0.9589   -0.9258   -0.8835   -0.8323
-0.7728

  Columns 56 through 63

  -0.7055   -0.6313   -0.5507   -0.4646   -0.3739   -0.2794   -0.1822   -0.0831

>> plot (x, y)
```




```
>> hold on
>> x ;
>> y1= cos(x) ;
>> plot (x, y1)
```

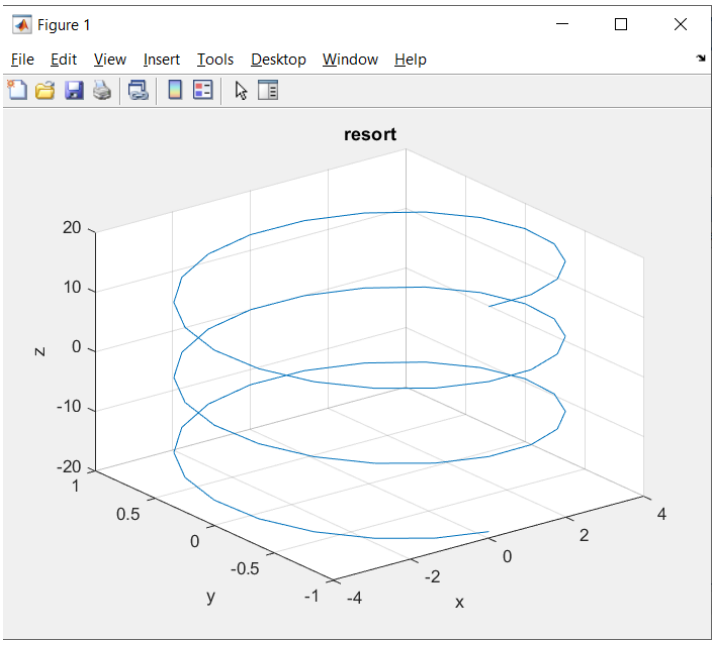


```
>> x= 0 : 0.1 :2*pi ;
>> sin (x) ;
>> plot (x, sin(x), 'r')

>> subplot (4, 2, 4)

>> plot (x, sin(x))
```



- **Graphique 3D :**

```
>> t= -3*pi : pi/10 : 3*pi ;

>> x= 4*sin(t) ; y= cos(t) ; z= 2*t ;

>> plot3(x, y, z)

>> title( 'resort')

>> xlabel ('x'), ylabel ('y'), zlabel ('z'),

>> grid
```



3. Fichiers et fonctions

Un script sous MATLAB est un fichier de commandes avec l'extension .m qui contient une suite de commandes Matlab.

a) Utilisation des structures de contrôles et des boucles itératifs :, if, for, while

```
a= [0 -1 4 -6 9 -7]
b= []
c= []
for i=1 : length(a),
        if a(i) <0
                b= [b a(i)] ;
        else
                c= [c a(i)] ;
        end
end

b =

   -1   -6   -7

>> c

c =

    0    4    9
```

b) Fonctions avec variables d'entrée/sortie

- La syntaxe d'une fonction Matlab est:

```
function [args1,args2,...] = nomfonction(arge1,arge2,...)
        instructions
end
```

args1,args2,... : sont les arguments de sortie de la fonction et peuvent être de n'importe quel type

arge1,arge2,... : sont les arguments d'entrée de la fonction et peuvent être de n'importe quel type

instructions : est un bloc d'instructions quelconque devant affecter les arguments de sortie *args1,args2,...*

- Lorsqu'il n'y a qu'un seul argument de sortie, on peut utiliser la syntaxe plus simple :

```
function args = nomfonction(arge1,arge2,...)
```

- Pour appeler une fonction on utilise de code suivant :

```
[vars1,vars2,...] = nomfonction(vare1,vare2,...)
```

Remarque : Une fonction doit être enregistrée dans un fichier **.m** du même nom que la fonction.

Exemple 1: Enregistrer la fonction ***fact*** avec paramètre d'entrée ***n***, dans le fichier « **fact.m** »:

```
function f= fact(n)
        f=1 ;
        for i=1 : n
                f= f*i ;
        end
end
```

- Exécution

```
>> fact(5)
fact(5)
ans =
  120
```

Exemple 2: Cas d'une fonction avec plusieurs arguments de sortie :

```
function [x, y]= calcul (a, b)
        x= a+b ;
        y= a*b ;
end
```

- Exécution :

```
>> [u, v]= calcul (2, 3)
u =
    5
v =
    6
```

Ministère de l'Enseignement Supérieure et de la Recherche Scientifique
Université de Sfax
Institut Supérieur de Biotechnologie de Sfax

Traitement numérique de signal médical

TP 2 : Les images sous MATLAB

2018 – 2019

3éme année Licence appliquée en Génie Biomédicale

Enseignants :

Nessrine ELLOUMI, Assistante Vacataire à ISBS de Sfax

Pr. Mohamed Salim BOUHLEL, Professeur à ISBS de Sfax

TP 2 : Les images sous MATLAB

Objectifs :

- Introduire les principaux types d'images utilisés par MATLAB
- Afficher les images

1.1 Type d'images sous MATLAB

Il existe quatre types d'images sous MATLAB :

- Images indexées
- Images d'intensité
- Images binaires
- Images RVB (Rouge Vert Bleu)

Sous MATLAB une image 2D est représentée par une matrice entière « M ».

Les éléments de cette matrice M[i, j] représente les pixels de l'image.

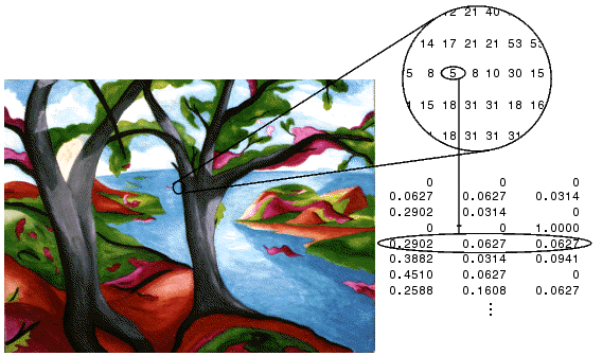
Soit une matrice P contienne les valeurs des pixels de l'image « photo1.jpeg »

P(5,7) représente la valeur du pixel de la ligne 5 et la colonne 7 de l'image P.

1.1.1 Images indexées

Une image indexée est composée de deux éléments : une palette de couleur (appelée *look-up table*) et une matrice de l'image.

Le couleur d'un pixel de l'image est représenté par un indice qui correspond à une valeur de la carte de couleur.



Exercice :

% Lire l'image trees

```
>>[I, LUT] = imread('trees.tif');
```

% Afficher cette image :

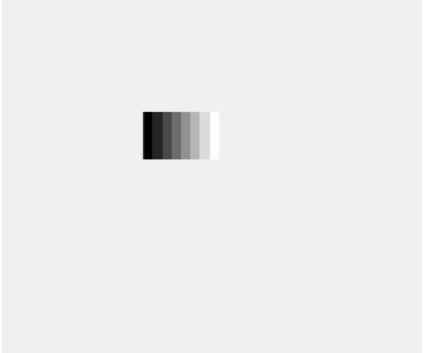
```
>>imshow(I)
```

1.1.2 Images d'intensité

Une image d'intensité est représentée par une matrice appelé matrice d'intensité. Les valeurs de cette matrice allant de 0.0 à 1.0. Ces valeurs représentent le niveau de gris ou une intensité de chaque pixel. 0.0 désigne le noir et 1.0 désigne le blanc.

Exemple :

```
X=ones(5,1)*[(0 :7)/7] ;
Imshow(X) ;
```



Exercice:

% Charger l'image **trees**

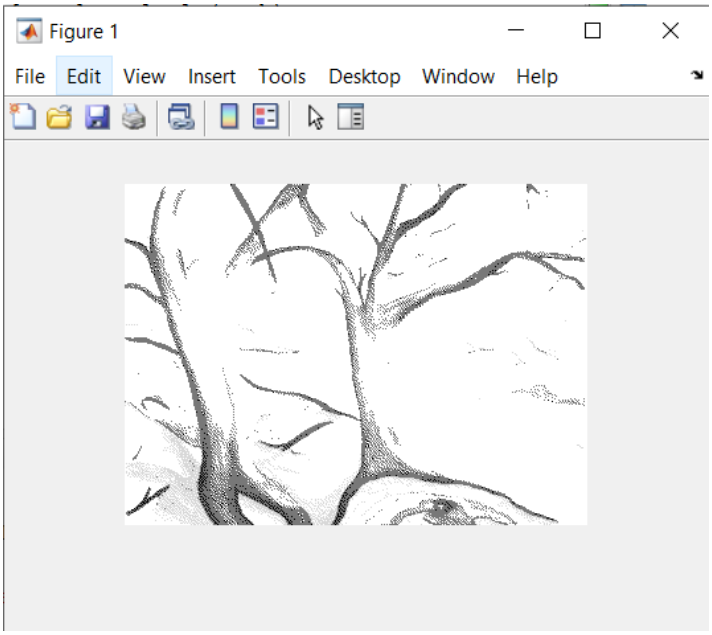
```
>>load trees
```

% convertir l'image indexée en image d'intensité

```
>>A=ind2gray(I, LUT);
```

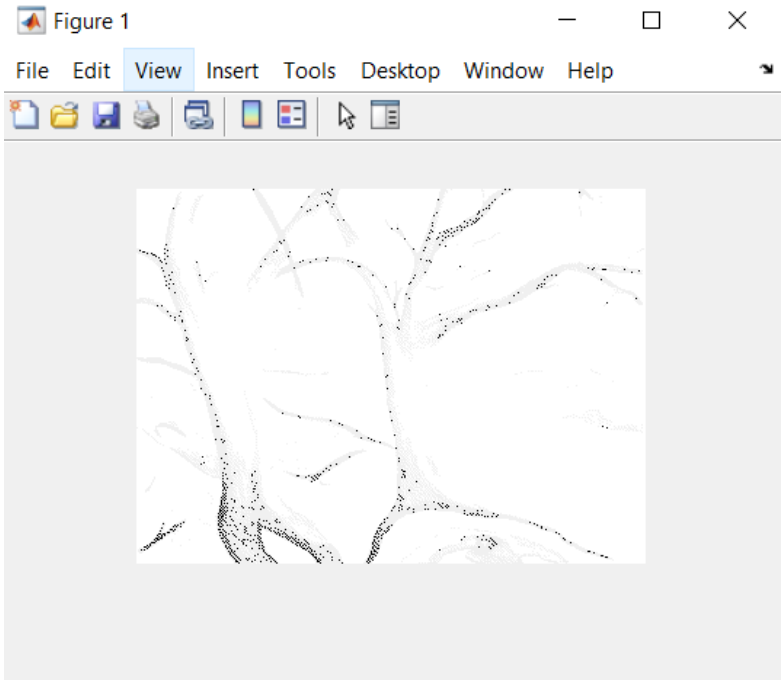
% afficher l'image indexée dans l'intervalle [0,32]

```
>>imshow (A,[0,32])
```

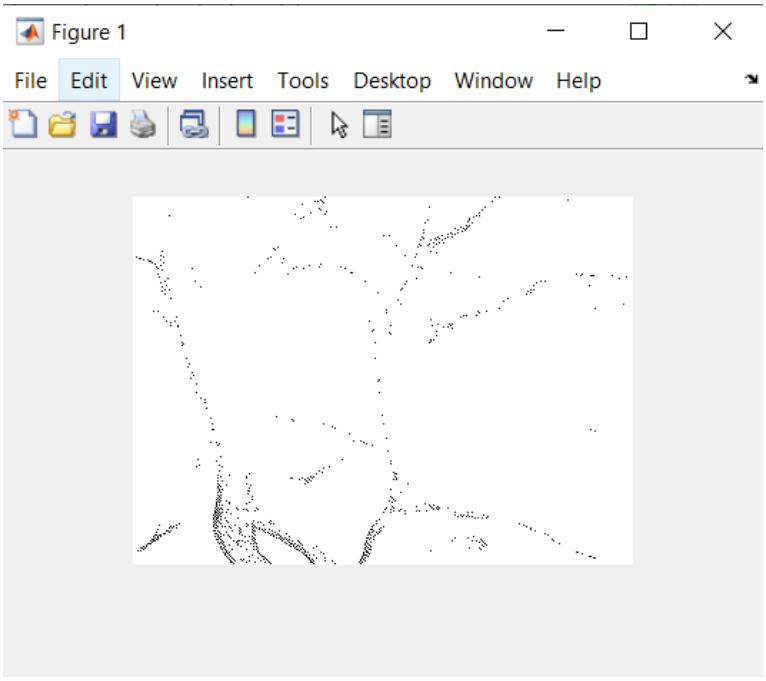


- **Charger et afficher l'image "trees" selon 16 et 4 niveaux de gris.**

16 niveaux de gris



4 niveau de gris



1.1.3 Image binaire

Une image binaire est représentée par une matrice qui contient deux niveaux de gris : 0 désigne la couleur noir et 1 désigne la couleur noir blanc.

Exercice :

% charger l'image trees dans la mémoire

```
>>load trees
```

% convertir l'image indexée en image d'intensité

```
>>I=ind2gray(I, LUT) ;
```

% détecter les contours

```
>>BW=edge(I);
```

% afficher l'image binaire inversée des contours

```
>>imshow(~BW,2)
```

1.1.4. Image RGB

Une image RGB est représentée par une matrice de taille (l,c,3). Chaque élément de cette matrice représente l'intensité de chaque pixel en Red, Green et Blue.

Soit une matrice Col :

Col (30,20,1) représente l'intensité en rouge.

Col (30,20,2) représente l'intensité en vert.

Col (30,20,3) représente l'intensité en bleu.

Exemple : Exécuter le script suivant :

```
close all
map=jet(64);
RGB=ones(100,64,3);
 for j=1:64
    for i=1:100
      RGB(i,j,1:3)=map(j,1:3);
```

```
    end
end
```

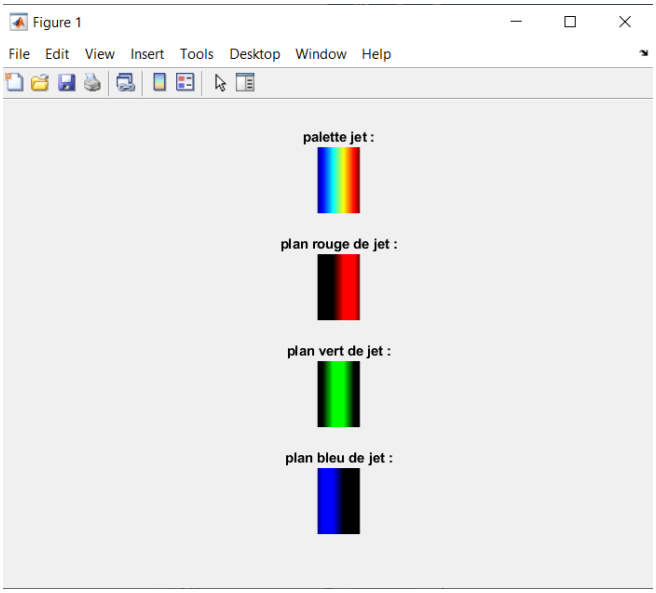
```
subplot(4,1,1) ;
imshow(RGB) ;
title('palette jet :');
```

```
rouge=RGB;
rouge(: , : , 2)=0;
rouge(: , : , 3)=0;
subplot(4,1,2) ;
imshow(rouge) ;
title('plan rouge de jet :') ;
```

```
vert=RGB;
vert(:,:,1)=0;
vert(:,:,3)=0;
subplot(4,1,3) ;
imshow(vert) ;
title('plan vert de jet :') ;
```

```
bleu=RGB;
bleu(:,:,1)=0;
bleu(:,:,2)=0;
subplot(4,1,4) ;
imshow(bleu) ;
title('plan bleu de jet :') ;
```

% La fonction ***jet*** sous Matlab crée un vecteur de n couleurs du bleu foncé, cyan, vert, jaune et rouge, jusqu'a le rouge foncé.



1.2 Affichage d'une image

1.2.1 Affichage simple

L'affichage simple permet d'afficher une image dans une figure.

- Pour afficher une image indexée :

```
>>Imshow(I, LUT)
```

I est la matrice de l'image. ***LUT*** est la carte de couleurs.

- Pour afficher une image d'intensité :

```
>>Imshow(I,[min,max])
```

I est la matrice d'image. ***[min, max]*** est l'intervalle du niveau de gris.

- Pour afficher une image binaire :

```
>>Imshow(BW,2)
```

BW est la matrice affichée en noir et blanc. Le pixel de valeur 0 est affiché en noir et celui de valeur 1 en blanc.

- Pour inverser le noir et le blanc :

```
>>Imshow(~BW,2)
```

1.2.2. Affichage multiple

L'affichage multiple permet d'afficher plusieurs images dans une seule figure.

Exemple :

```
load trees
subplot(1,2,1) ,imshow(X,map), title('Image arbres');
load clown
subplot(1,2,2),subimage(X,map), title('Image clown') ;
axis off
```

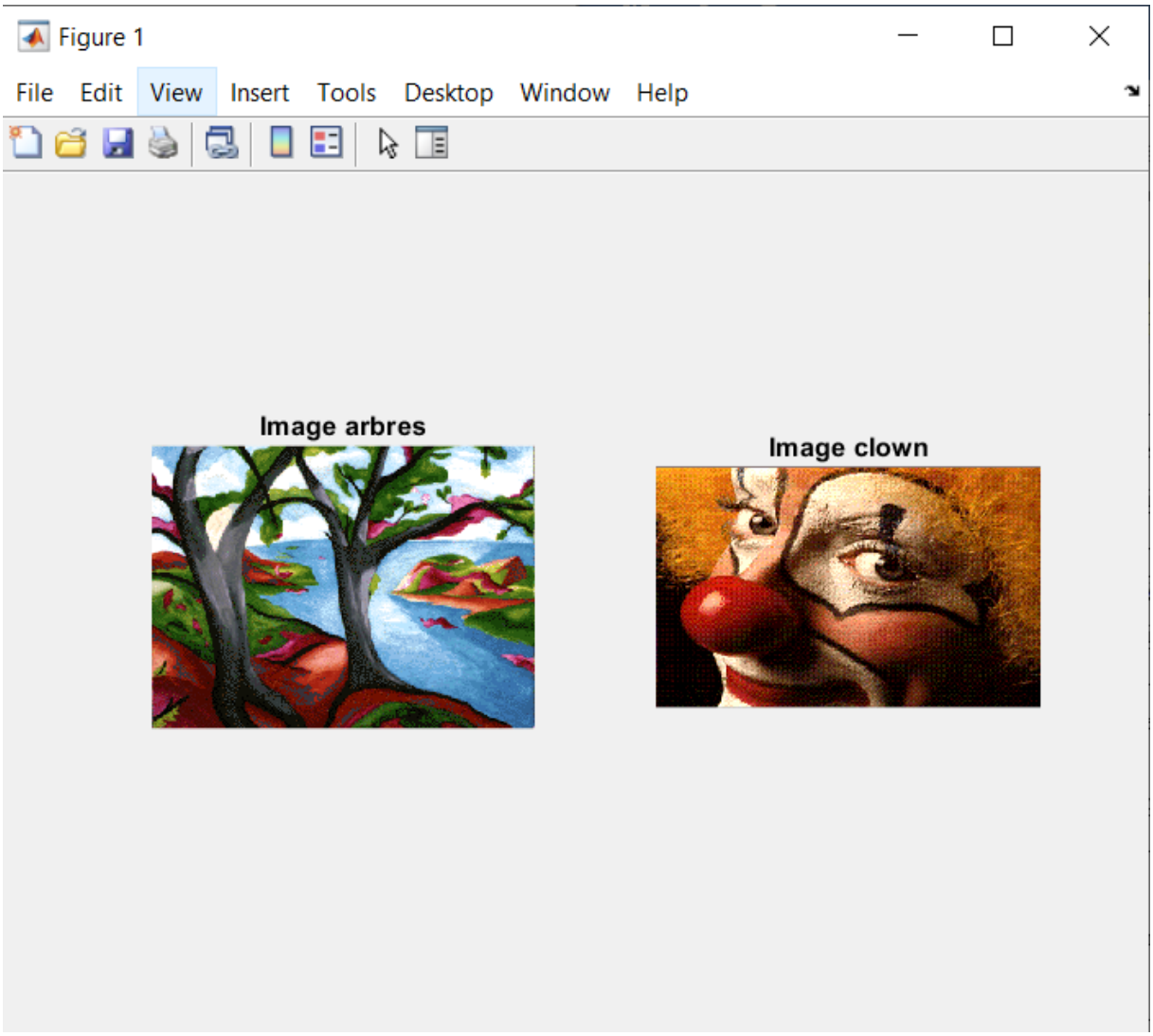


Figure 1
File Edit View Insert Tools Desktop Window Help
Image arbres
Image clown

Ministère de l'Enseignement Supérieure et de la Recherche Scientifique
Université de Sfax
Institut Supérieur de Biotechnologie de Sfax

Traitement numérique de signal médical

TP 3 : Compression d'image (JPEG)

2018 - 2019

3éme année Licence appliquée en Génie Biomédicale

Enseignants:

Nessrine ELLOUMI, Assistante Vacataire à ISBS de Sfax

Pr. Mohamed Salim BOUHLEL, Professeur à ISBS de Sfax

TP 3 : Compression d'image JPEG

Objectifs :

La compression d'image permet de réduire la redondance des données d'une image afin de minimiser l'espace de stockage ou la transmettre sur un réseau à faible bande passante.

La norme J.P.E.G (Joint Photographic Experts Groups) permet de compresser des images au format RGB avec un minimum de distorsion.

Manipulation

Créer un fichier ***TP3.m*** pour réaliser le travail suivant :

1) Lire l'image "*horse.bmp*" (l'affecter à la variable ***Im_bmp***) en utilisant la commande ***imread***.

2) Compresser l'image ***Im_bmp.bmp*** en une image jpeg (***Im_out.jpg***) avec la commande suivante :

```
imwrite(Im_bmp, 'Im_out.jpg','jpg','quality', Q); Q=10 ;
```

3) Lire l'image " ***Im_out.jpg***" (l'affecter à la variable ***Im_jpg***).

```
Im_jpg=imread('Im_out.jpg')
```

4) Extraire l'image différence ***diff*** entre ***Im_bmp*** *et* ***Im_out.***

```
Diff=Im_bmp-Im_out
```

5) Afficher l'image ***diff*** dans unefigure (figure 1).

```
imshow(diff)
```

6) Calculer le MSE (l'erreur quadratique moyenne) entre ***Im_bmp*** *et* ***Im_out.***

$$MSE = \frac{1}{N} \sum_{p=1}^{N} (Im1 - Im2)^2$$

7) Calculer le PSNR entre ***Im_bmp*** et ***Im_out.***

$$PSNR = 10\log\left(\frac{X_{max}^2}{MSE}\right)$$

Avec X_{max} désigne la luminance maximale possible.

psnr(Im_bmp,Im_out)= 23.0656

8) Déterminer la taille des images ***Im_bmp.bmp*** et ***Im_out.jpg.***

size(Im_bmp)

ans =

186 271 3

size(Im_bmp)

ans =

186 271 3

Ministère de l'Enseignement Supérieure et de la Recherche Scientifique
Université de Sfax
Institut Supérieur de Biotechnologie de Sfax

Traitement numérique de signal médical

TP 4 : Opérateurs Morphologiques

2018-2019

3éme année Licence appliquée en Génie Biomédicale

Enseignants:

Nessrine ELLOUMI, Assistante Vacataire à ISBS de Sfax

Pr. Mohamed Salim BOUHLEL, Professeur à ISBS de Sfax

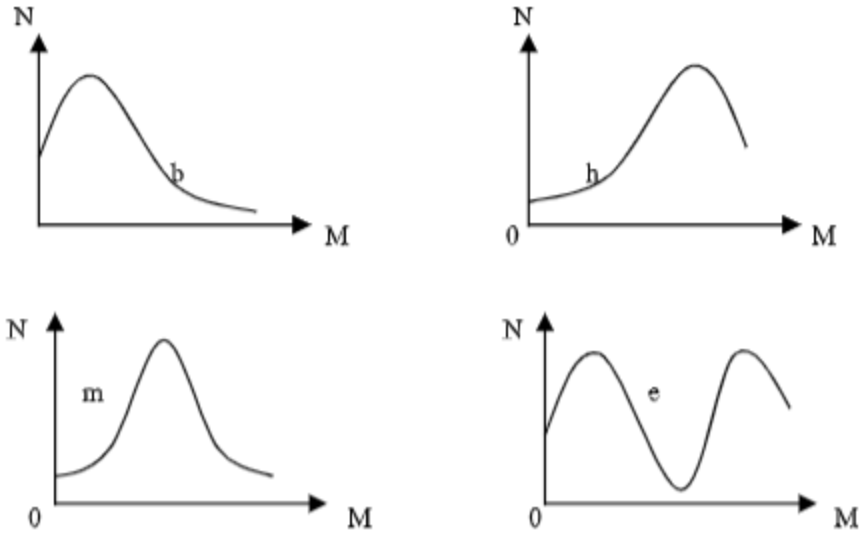
TP 4 : Opérateurs Morphologiques

Objectifs :

- Manipulation de l'histogramme d'une image.
- Appliquer des opérations Morphologique sur des images 2D.

Histogramme d'une image

L'histogramme d'une image est la distribution des intensités dans cette image.
La fonction **imhist(I,n)** sous MATLAB calcule l'histogramme de l'image I en utilisant n niveaux de gris.
L'histogramme d'une image permet de détecter les défauts de contraste d'une image.



- L'histogramme b est concentré sur les bas niveaux de gris, l'image est trop sombre.
- L'histogramme h est concentré sur les hauts niveaux de gris, l'image est trop claire.
- L'histogramme m est concentré sur les niveaux de gris moyens, l'image manque de contraste.
- L'histogramme e est concentré sur les niveaux extrêmes, l'image manque de nuances intermédiaires entre le sombre et le clair.

Egalisation de l'histogramme

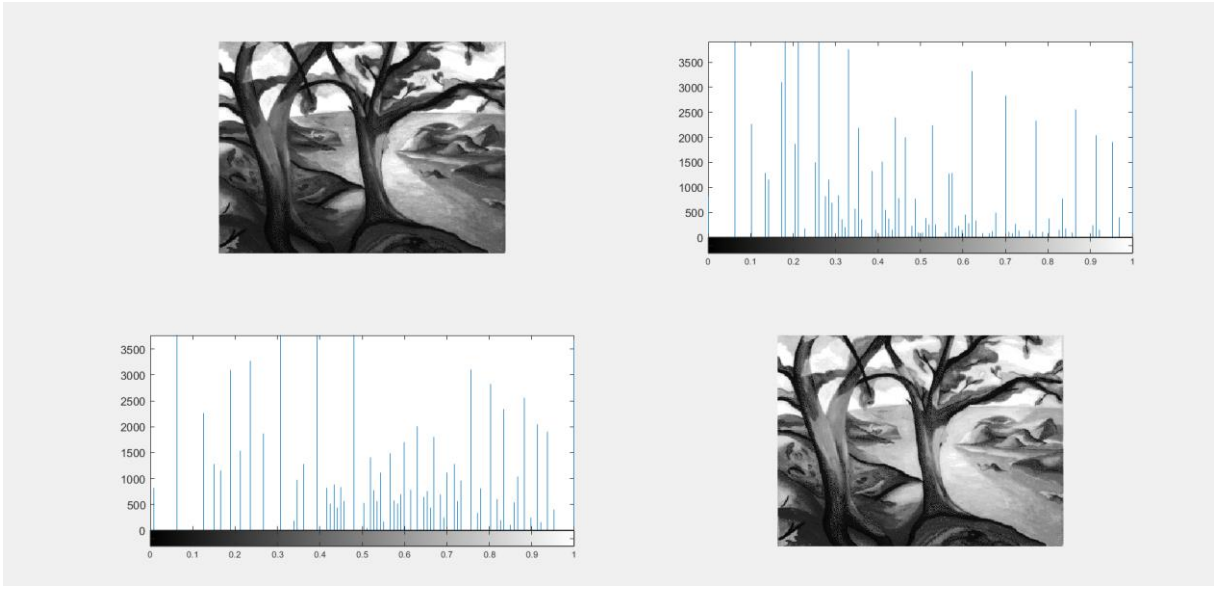
L'égalisation de l'histogramme permet de distribuer uniformément le nombre de pixel selon les niveaux de gris de l'image.

Sous MATLAB la fonction **histeq(I,n)** effectue l'égalisation avec I représente la matrice de l'image et n est le nombre de niveau de gris.

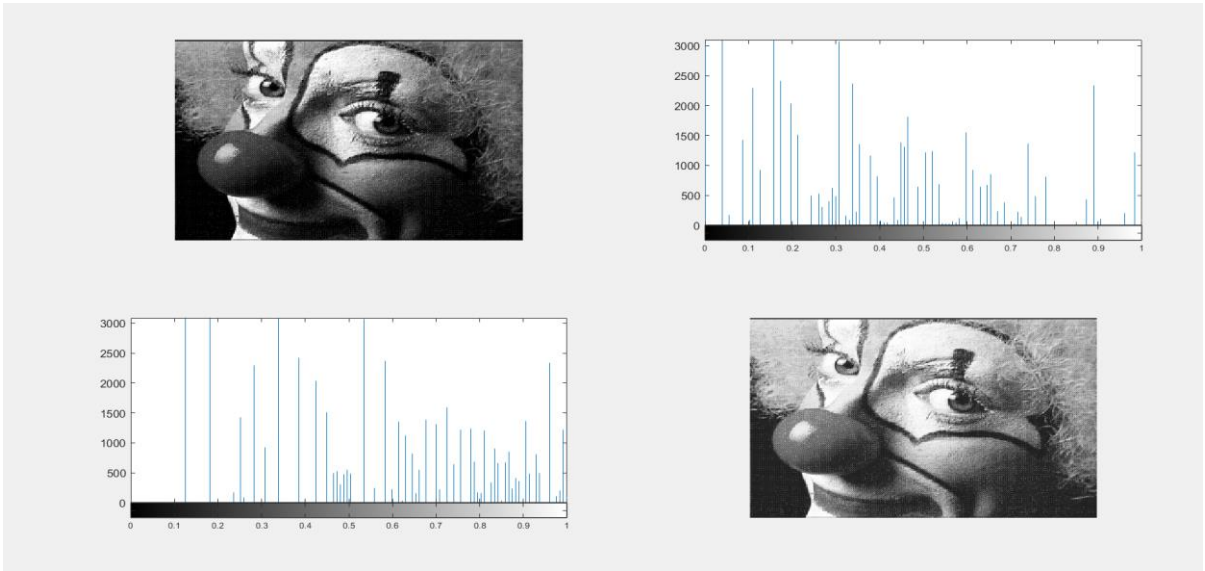
Charger l'image trees dans la même figure :

- Afficher l'image trees répartie sur 128 niveaux de gris
- Afficher l'histogramme correspondant
- Afficher l'histogramme égalisé sur 128 niveaux de gris
- Afficher l'image dont l'histogramme est égalisé

```
load trees
I=ind2gray(X,map) ;
subplot(2,2,1),imshow(I)
subplot(2,2,2),imhist(I,128)
I=histeq(I,128)
subplot(2,2,3),imhist(I,128)
subplot(2,2,4),imshow(I)
```



Refaire le même travail pour l'image clown



Opérateur morphologique

- *Dilatation/Erosion*

La dilatation permet la disparition de certains trous et la connexion de certains éléments relativement à la taille de l'élément structurant appliqué. Le résultat d'une dilatation est une structure plus grande que la structure d'origine.
L'érosion permet la disparition et le gommage des particules blanches plus petites que l'élément structurant.

- *Ouverture/Fermeture*

Une fermeture morphologique est l'application d'une dilatation suivie par une érosion.

Une ouverture morphologique est l'application d'une érosion suivie par une dilatation.

Manipulation

Créer un fichier ***TP4.m*** pour réaliser le travail suivant :

1) Lire l'image "*peppers.png* ".

```
>>I=imread('peppers.png');
```

2) Transformer l'image en niveau de gris.

```
>>x=rgb2gray(I);
```

3) Filtrer l'image en utilisant la commande matlab *imfilter* pour le filtrage et *ones* pour la création du masque de taille (3,3).

```
>>mas=ones(3,3)*1/9
>>IF=imfilter(x,mas);
>>imshow(IF)
```

4) Afficher l'image origine, l'image filtrée et leurs histogrammes dans une même figure.

```
>>subplot(2,2,1),imshow(x)
>>subplot(2,2,2),imhist(x)
>>subplot(2,2,3),imshow(IF)
>>subplot(2,2,4),imhist(IF)
```

5) Appliquer une égalisation d'histogramme sur l'image filtrée *histeq(img,r)*.

```
>>IFH=histeq(IF,4);
>>xH=histeq(x,4);
```

6) Afficher l'image filtrée et son histogramme avant et après l'égalisation d'histogramme dans une même figure.

```
>>subplot(2,2,1),imshow(xH)
>>subplot(2,2,2),imhist(xH)
```

```
>>subplot(2,2,3),imshow(IFH)
>>subplot(2,2,4),imhist(IFH)
```

7) Appliquer une Binarisation à l'image obtenue.

```
[l,c]=size(xH);s=127;
for i=1:l
      for j=1:c
        if xH(i,j)<s
            xB(i,j)=0;
        else
            xB(i,j)=255;
        end
    end
end
```

8) Appliquer une Erosion puis une Dilatation sur l'image binaire.

Erosion:

se = strel('disk',2);

Imerode = imerode(img,se);

```
>>se = strel('disk',2);
>>Imerode = imerode(x,se)
```

Dilataion: ***Imdilate = imdilate(Imerode,se)***

```
>>Imdilate = imdilate(Imerode,se);
>>subplot(2,2,1),imshow(Imdilate);
>>subplot(2,2,2),imshow(Imerode);
```

9) Détecter le contour de l'image binaire en utilisant la commande matlab *edge* en appliquant le filtre de Canny.

```
>>cont=edge(xH,'canny');
>>imshow(cont);
```

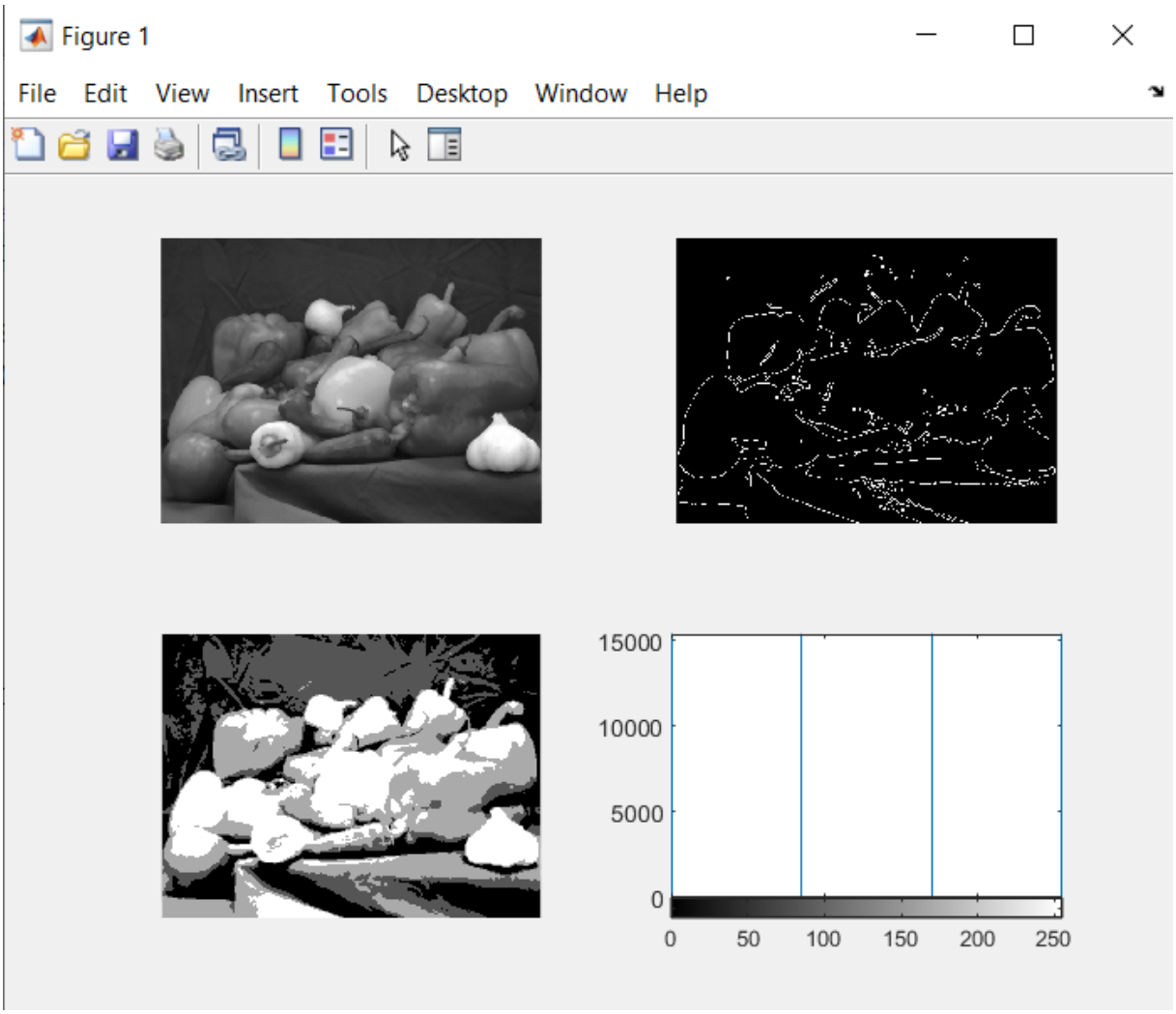


Figure 1
File Edit View Insert Tools Desktop Window Help
15000
10000
5000
0
0 50 100 150 200 250