



## Codage source :

COMPTE RENDUE : PROJET

---

*Encadré par :*

*MME. Malika ALAMI MARKTANI*

---

## FIGURE 1 :

Front :

Description :

C'est la 1<sup>ère</sup> figure apparue lors du démarrage de l'application , elle contient :

- Un texte : pour la demande du type de donnée qu'on va coder
- Deux radio boutons : pour que l'utilisateur effectue un choix entre image et texte
- Un bouton : pour valider le choix

code :

pour initialiser les radio boutons j'ai créer cette fonction :

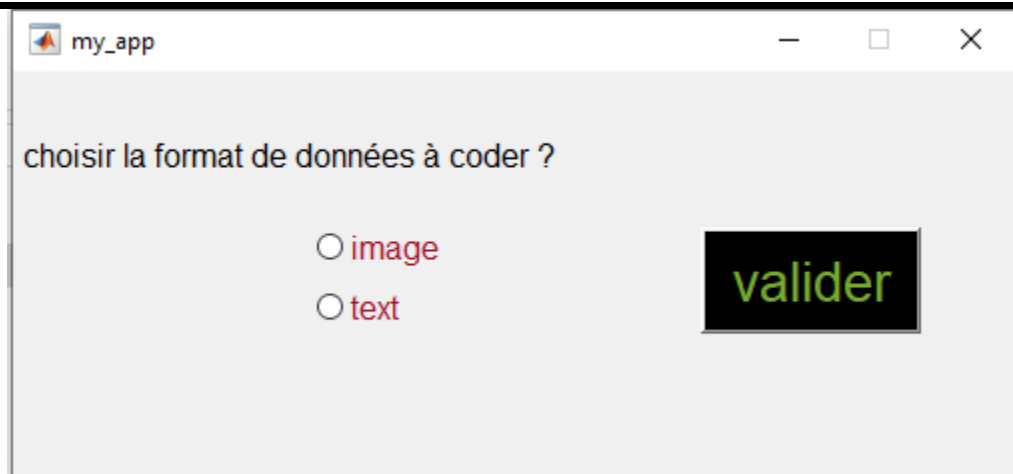
```
function radio_off(handles)  
    set(handles.text_edit, 'value', 0)  
    set(handles.image_edit, 'value', 0)
```

j'ai modifié dans la fonction de clique sur le bouton 'valider' :

- Si l'utilisateur choisira l'image comme type d'entrée on va basculer vers une autre figure pour coder l'image sinon on ouvre la figure de traitement de texte

```
% --- Executes on button press in validation_button.  
function validation_button_Callback(hObject, eventdata, handles)  
% hObject    handle to validation_button (see GCBO)  
% eventdata  reserved - to be defined in a future version of MATLAB  
% handles    structure with handles and user data (see GUIDATA)  
text = get(handles.text_edit, 'value')  
image = get(handles.image_edit, 'value')  
if text == 1  
    close(my_app)  
    my_app_fig_txt  
end  
if image == 1  
    close(my_app)  
    my_app_img  
end  
end
```

Figure complète :

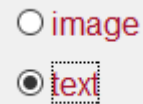


## FIGURE 2 :

Figure du texte :

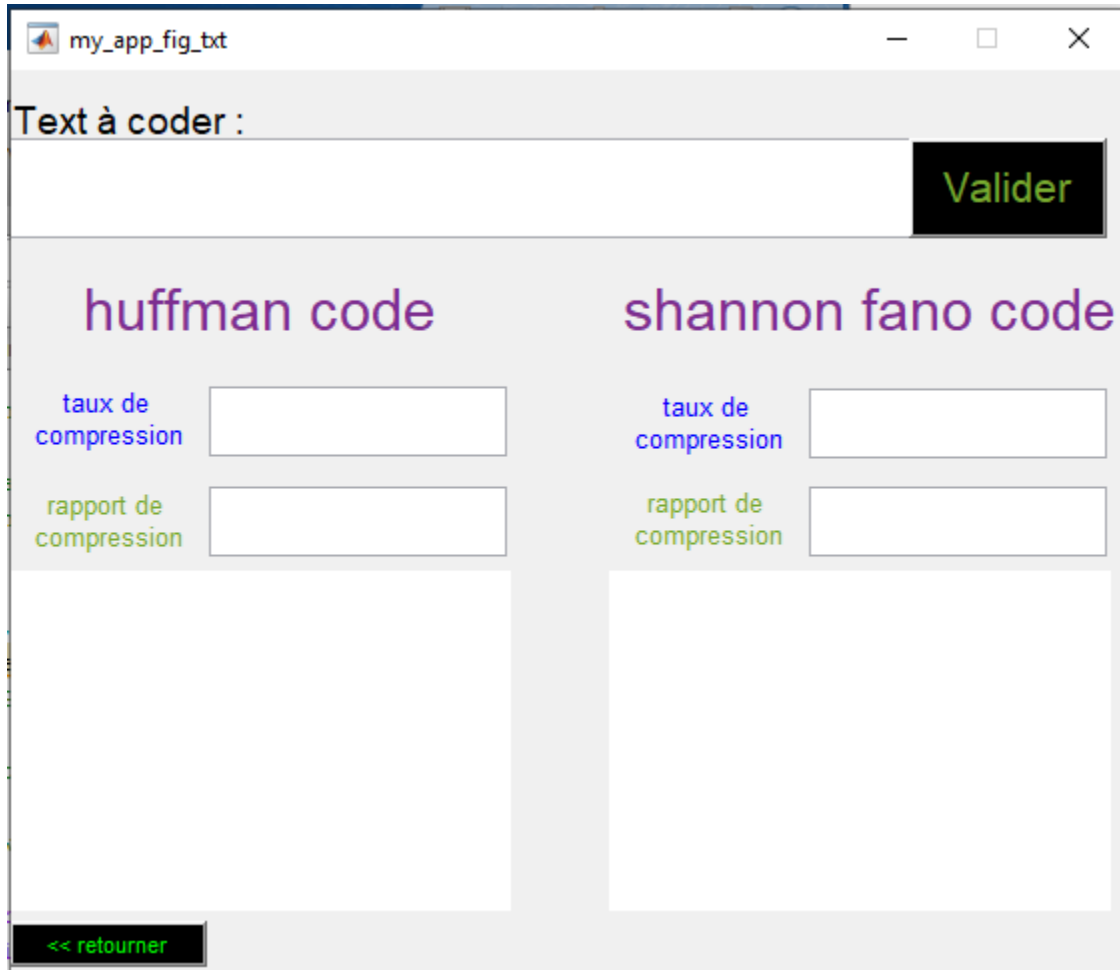
Description :

Cette fenêtre est apparue lorsqu'on valide le checkbox du texte dans la figure precedente , elle contient :



- Un texte : pour la demande d'entrer du texte
- Une zone de texte : pour entrer le texte à coder
- Un bouton : pour valider le texte et afficher les resultats de calcule
- 2 régions (une pour Hoffman code et l'autre pour Shannon Fano code)
  - *Hoffman* :
    - Zone de texte pour affichage de taux de compression
    - Zone de texte pour affichage de rapport de compression
    - Zone de texte pour affichage du texte codé en binaire
  - *Shannon Fano*
    - Zone de texte pour affichage de taux de compression
    - Zone de texte pour affichage de rapport de compression
    - Zone de texte pour affichage du texte codé en binaire
- Un bouton pour retourner a la figure d'accueil .

Figure complète :



code :

on va se baser sur les fonctions développées lors des TPs :

*myapp\_shannon\_fano.m* :

qui va prendre comme entré le texte , et le coder en Shannon Fano, puis calculer le taux de compression et le rapport de transmission pour ce texte :

```
function [bin,taux_c,rapport_c]=myapp_shannon_fano(text)
%bin est le text codee en binaire
bin=''
SIG = text
symboles=unique(SIG)
length(symboles);
%find proba
for i=1:length(symboles)
    k=findstr(SIG,symboles(i))
    n=length(k)
    Pr(i)=n/length(SIG)
end
Pr
symboles=double(symboles);
m_code=ShannonFanoCode(symboles,Pr)
code2=EntropyEncoder(SIG,m_code)
[n, m] = size(code2)
s=' '
for i=1:m
    bin= [bin,s,num2str(code2{1,i})]
end
#####taux de compression#####
taux_c = length(code2) / (length(SIG)*8);
#####Rapport de compression#####
rapport_c = ((length(SIG)*8)-length(code2)) / (length(SIG)*8)
end
```

*myapp\_huffman.m :*

qui va prendre comme entrer le texte , et le coder en Hoffman, puis calculer le taux de compression et le rapport de transmission pour ce texte

```
function [bin,taux_c,rapport_c]= myapp_huffman(text)
%bin est le text codée en binaire
bin=''
SIG = text
symboles=unique(SIG)
length(symboles);
%find proba
for i=1:length(symboles)
    k=findstr(SIG,symboles(i))
    n=length(k)
    Pr(i)=n/length(SIG)
end
Pr
symboles=double(symboles);
m_code=huffmandict(symboles,Pr);
code2=huffmanenco(SIG,m_code);
[n, m] = size(code2)
s=' '
for i=1:m
    bin= [bin,s,num2str(code2(1,i))];
end
#####taux de compression#####
taux_c = length(code2) / (length(SIG)*8);
#####Rapport de compression#####
rapport_c = ((length(SIG)*8)-length(code2)) / (length(SIG)*8)
end
```

*myapp\_huffman\_shannon.m :*

qui va prendre comme entré le texte , et exécuter les deux fonction au dessus pour bien retourner une string indiquant lequel des deux algorithmes est plus performant pour le texte entré

```
function [code1,code2,s,tcl,rc1,tc2,rc2] = myapp_huffman_shannon(text)

[code1,tcl,rc1] = myapp_huffman(text)
[code2,tc2,rc2] = myapp_shannon_fano(text)

if (tcl>tc2) s='algorithm du codage de shannon fano est plus performant que celui de huffman'
elseif (tcl<tc2) s='algorithm du codage de huffman est plus performant que celui de shannon fano'
else s='algorithm du codage de shannon fano et celui de huffman ont la meme performance'
end
end
```

*my\_app\_fig\_txt.m*

j'ai modifier dans la fonction de click sur le bouton valider :

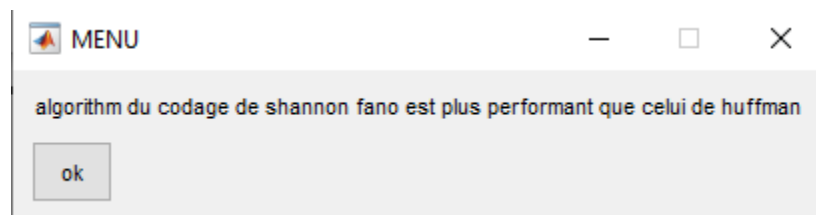
elle va prendre le texte le passer dans la fonction de calcul  
« myapp\_huffman\_shannon » et elle va retourner 7 paramètres :

- le codage du texte en binaire de Shannon Fano et de Huffman
- le taux de compression et rapport de compression des deux codes
- le résultat de comparaison en se basant sur le taux de compression

puis on va les afficher chacun dans la zone qui lui correspond.

```
% --- Executes on button press in validation_button.
function validation_button_Callback(hObject, eventdata, handles)
% hObject    handle to validation_button (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
txt = get(handles.enter_text_edit, 'string')
[code1, code2, result, tc1, rc1, tc2, rc2] = myapp_huffman_shannon(txt)
set(handles.huffman, 'string', code1)
set(handles.shannon, 'string', code2)
set(handles.taux1, 'string', tc1)
set(handles.rapport1, 'string', rc1)
set(handles.taux2, 'string', tc2)
set(handles.rapport2, 'string', rc2)
menu(result, 'ok')
```

Avec un message pop-up indiquant le plus performant des deux codages :



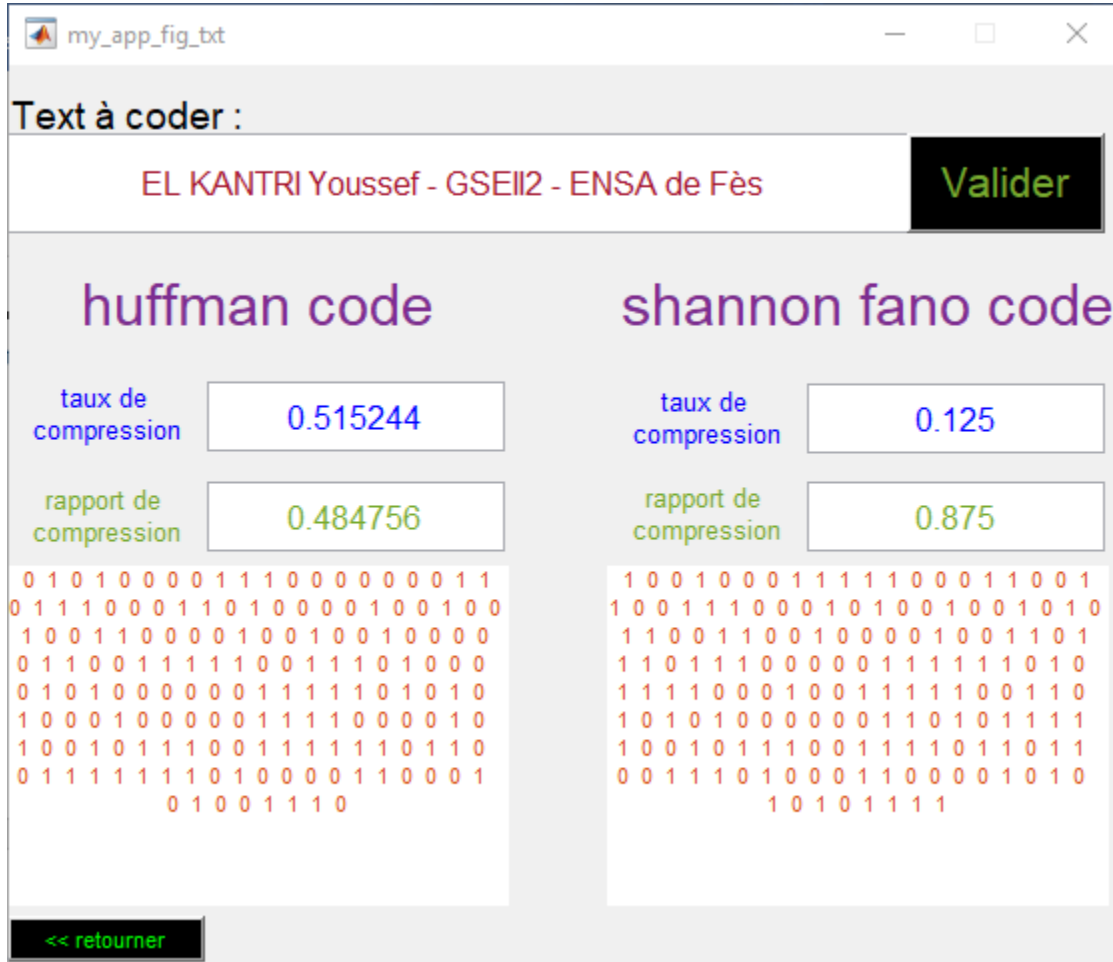
Pour retourner à la figure d'accueil on va configurer la fonction du click sur le bouton : « retourner » :

```
% --- Executes on button press in retour.
function retour_Callback(hObject, eventdata, handles)
% hObject    handle to retour (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close(my_app_fig_txt)
my_app
```



## Résultat :

On va tester l'algorithme on entrant le texte « EL KANTRI Youssef - GSEII2 - ENSA de Fès »



Text à coder :

EL KANTRI Youssef - GSEII2 - ENSA de Fès

Valider

### huffman code

taux de compression: 0.515244

rapport de compression: 0.484756

```

0 1 0 1 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 1 1
0 1 1 1 0 0 0 1 1 0 1 0 0 0 0 1 0 0 1 0 0
1 0 0 1 1 0 0 0 0 1 0 0 1 0 0 1 0 0 0 0
0 1 1 0 0 1 1 1 1 1 0 0 1 1 1 0 1 0 0 0
0 1 0 1 0 0 0 0 0 0 1 1 1 1 1 0 1 0 1 0
1 0 0 0 1 0 0 0 0 0 1 1 1 1 0 0 0 0 1 0
1 0 0 1 0 1 1 1 0 0 1 1 1 1 1 1 0 1 1 0
0 1 1 1 1 1 1 1 0 1 0 0 0 0 1 1 0 0 0 1
      0 1 0 0 1 1 1 0
  
```

### shannon fano code

taux de compression: 0.125

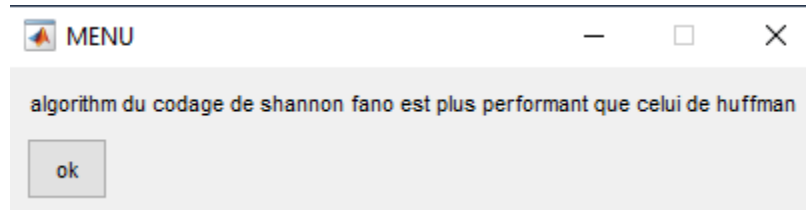
rapport de compression: 0.875

```

1 0 0 1 0 0 0 1 1 1 1 1 0 0 0 1 1 0 0 1
1 0 0 1 1 1 0 0 0 1 0 1 0 0 1 0 0 1 0 1 0
1 1 0 0 1 1 0 0 1 0 0 0 0 1 0 0 1 1 0 1
1 1 0 1 1 1 0 0 0 0 0 1 1 1 1 1 1 0 1 0
1 1 1 1 0 0 0 1 0 0 1 1 1 1 1 0 0 1 1 0
1 0 1 0 1 0 0 0 0 0 0 1 1 0 1 0 1 1 1 1
1 0 0 1 0 1 1 1 0 0 1 1 1 1 0 1 1 0 1 1
0 0 1 1 1 0 1 0 0 0 1 1 0 0 0 1 0 1 0
      1 0 1 0 1 1 1 1
  
```

<< retourner

Avec un message pop-up :



MENU

algorithm du codage de shannon fano est plus performant que celui de huffman

ok

### FIGURE 3 :

#### Codage de l'image :

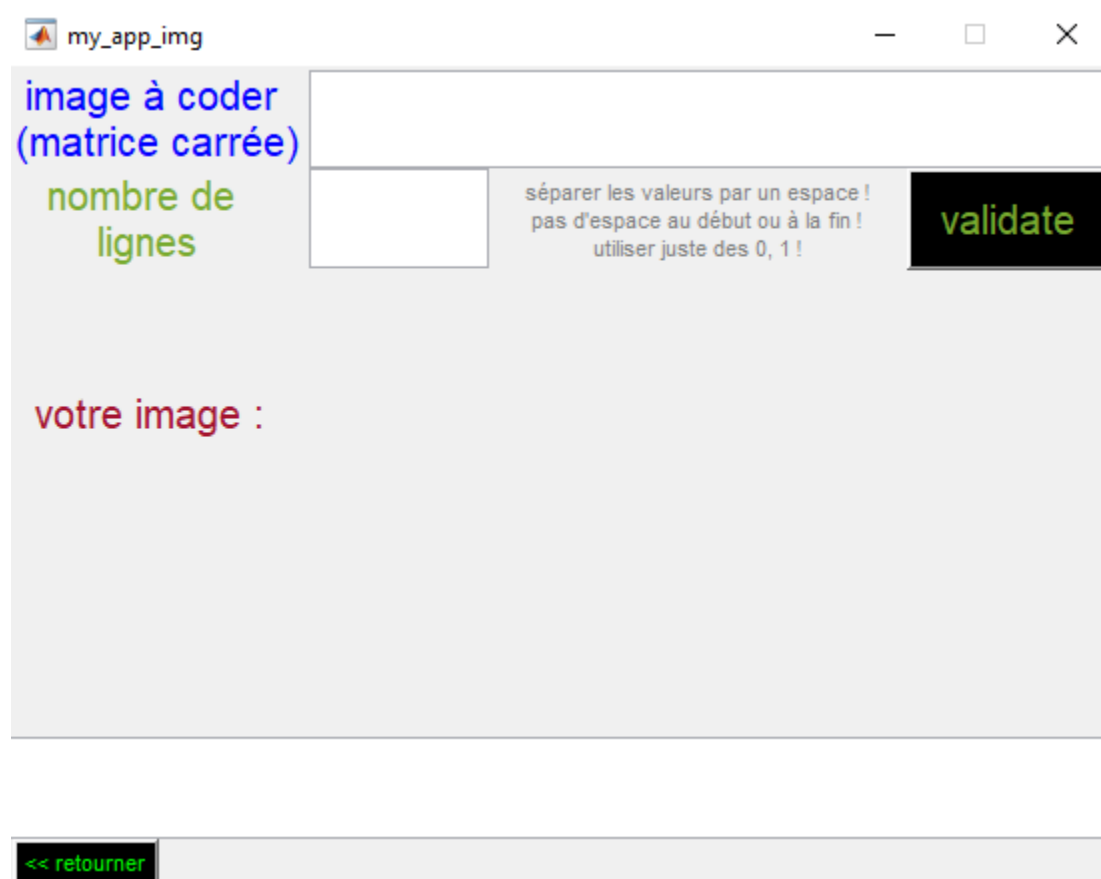
#### Description :

Cette fenêtre est apparue lorsqu'on valide le checkbox d'image dans la figure d'accueil précédente, elle contient :



- Un texte : pour la demande d'entrer la matrice
- Une zone de texte : pour entrer la matrice à coder sous forme de 0 - 1
- Un texte : pour la demande d'entrer la taille de la matrice ( le nombre de lignes)
- Une zone de texte : pour entrer la taille de la matrice
- Un bouton : pour valider la matrice entrée et afficher les résultats de calcul
- Un texte et un axe : pour afficher la matrice sous forme d'une image noire et blanc.
- Une zone de texte : indiquant la plus performante des 3 lectures : horizontal, vertical ou en zigzag en se basant sur le nombre des zéros successives.
- Un bouton pour retourner a la figure d'accueil .

## Figure complète :



## code :

on va se baser sur les fonctions développées lors des TPs :

*lect\_hor.m* :

qui va prendre comme entrée une matrice ,et retourne un vecteur contenant les valeurs de la matrice en lecture horizontale :

```
function s=lect_hor(X)
    s=[];
    for k=1:size(X,1)
        for j=1:size(X,2)
            s= [s X(k,j)];
        end
    end
end
```

*lect\_ver.m :*

qui va prendre comme entré une matrice ,et retourne un vecteur contenant les valeurs de la matrice en lecture vertical :

```
function s=lect_vert(X)
    s=[];
    for k=1:size(X,1)
        for j=1:size(X,2)
            s= [s X(j,k)];
        end
    end
end
```

*lect\_zigzag.m :*

qui va prendre comme entrée une matrice ,et retourne un vecteur contenant les valeurs de la matrice en lecture en zigzag :

```
function s=lect_zigzag(X)
    s=[];
    for a=1:size(X,1)
        for cond=[1:a;a:-1:1]
            i=cond(1);
            j=cond(2);
            if (mod(a,2)==1)
                s=[s X(j,i)];
            else
                s=[s X(i,j)];
            end
        end
    end
    for a=size(X,1)-1:-1:1
        for cond= [size(X,2)-a+1:size(X,2);size(X,2):-1:size(X,2)-a+1]
            i=cond(1);
            j=cond(2);
            if (mod(a,2)==1)
                s=[s X(j,i)];
            else
                s=[s X(i,j)];
            end
        end
    end
end
```

*number\_of\_zeros :*

qui va prendre comme entrée un vecteur ,et retourner le nombre de zeros successives dans ce vecteur .

```
function a=number_of_zeros(v)
    n=length(v) ;
    c=0 ;
    a=0 ;
    for i=1:n
        if (v(i)==0)
            c=c+1 ;
            a=max(a,c) ;
        else
            c=0;
        end
    end
end
```

*lecture.m*

qui va prendre comme entrée une matrice ,appliquer sur elle les 3 méthodes de lecture , et retourner une string indiquant le plus performants de ces derniers après la comparaison de nombre de zéros successive de chacun des 3 lectures .

```
function s=lecture(M)
    v= lect_vert(M)
    h= lect_hor(M)
    z= lect_zigzag(M)
    a=number_of_zeros(v)
    b=number_of_zeros(h)
    c=number_of_zeros(z)
    if (max([a,b,c])==a) s= strcat('la lecture vertical est performante, le nombre de zeros successives est : ',string(a))
    elseif (max([a,b,c])==b) s= strcat('la lecture horizontal est performante, le nombre de zeros successives est : ',string(b))
    else s= strcat('la lecture en zigzag est performante, le nombre de zeros successives est : ',string(c))
    end
end
```

*my\_app\_img.m*

j'ai modifier dans la fonction de click sur le bouton valider :

```
% --- Executes on button press in validation.
function validation_Callback(hObject, eventdata, handles)
% hObject    handle to validation (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles     structure with handles and user data (see GUIDATA)
text = get(handles.matrice_edit, 'string')
arr = str2double(split(text))
size = str2double(get(handles.size_edit, 'string'))
carre_size=size*size
if (carre_size == length(arr))
    matrice=[]
    for i=1:size
        for j=1:size
            matrice(i,j) = arr(((i-1)*size)+j);
        end
    end

    s=lecture(matrice)
    set(handles.lecture, 'string', s)
    axes(handles.axes1)
    imshow(matrice)
else
    error_matrix_size_msg
end
```

Elle va prendre le texte écrit dans la zone de la matrice et le transforme en une matrice en utilisant la fonction `split()` du Matlab , puis on récupère la taille de la matrice ; on la compare avec la taille calculée de notre tableau on utilisant la fonction `length()`,

- si elles sont égaux, on va afficher la matrice en tant qu'une image , avec un commentaire indiquant quelle lecture est la plus performante avec le nombre de zéros successives.
- sinon on affiche une petite fenêtre de type « modal question dialog » informant sur l'erreur commise, 'error\_matrix\_size\_msg.fig' .

Pour retourner a la figure d'accueil on va configurer la fonction du click sur le bouton : « retourner » :

```
% --- Executes on button press in retour.
function retour_Callback(hObject, eventdata, handles)
% hObject    handle to retour (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close(my_app_fig_txt)
my_app
```

## Résultat :

On va tester l'algorithme on entrant le suite de nombre suivante :

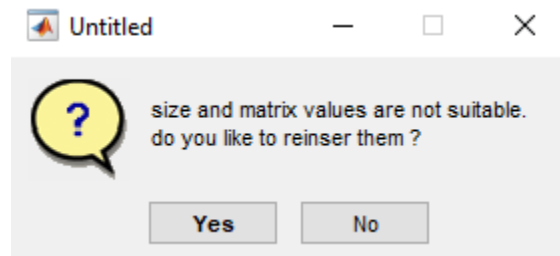
```
0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 1 0 0 0 1 1 0 1 0 1 1 0 0 1 0 0 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 0 0 1
0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

Avec un nombre de lignes = 9

- Si on respecte pas la taille de matrice ou ces conditions :

séparer les valeurs par un espace !  
pas d'espace au début ou à la fin !  
utiliser juste des 0, 1 !

on aura le message d'erreur :



- Si on respect ces conditions on aura comme résultat :

my\_app\_img

image à coder  
(matrice carrée)

nombre de  
lignes

9

séparer les valeurs par un espace !  
pas d'espace au début ou à la fin !  
utiliser juste des 0, 1 !

validate

votre image :



la lecture horizontale est performante, le nombre de zeros successives est :22

<< retourner

## CONCLUSION :

Après avoir réaliser ce projets , on a bien appris a programmer avec le GUIDE de Matlab avec ces différentes fenêtres et composantes , et on a bien compris les différents algorithmes de codage (Shannon Fano , Hoffman , RSA ...), c'est bien de pratiquer tels algorithmes avec telles méthodes pour clarifier les choses et se motiver à travailler.

*La connaissance théorique est un trésor dont la pratique est la clé*

*Thomas Fuller...*

PAGE 15