Université EUROMED de Fès
Ecole d'**I**ngénierie **D**igitale et d'**I**ntelligence **A**rtificielle

Programmation MATLAB

RAPPORT DE PROJET

Résolution du Système Linéaire(GUI)

Réalisé par:
Oumayma EL BAKKALI
Olaya LATOUBI

Encadré par:
Dr.Meryam EL MOUHTADI

Remerciments:

- Tout d’abord, nous tenons à remercier tout particulièrement et à témoigner toute notre reconnaissance à **Dr.Meryam EL MOUHTADI**, encadrante de projet, pour son dévouement et son soutient dans la concrétisation de ce projet.

Sommaire:

I. Présentation du projet:

1)-Notion générale:

- ❖ Avant de commencer nous allons faire un rappel sur les systèmes linéaires:

système linéaire, aussi appelé “système d’équations linéaires", est un système de telles équations ou les variables y apparaissent de manière séparées et toujours à la puissance 1.

Un système linéaire de n équations à p inconnues est un système du type:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1p}x_p & = b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2p}x_p & = b_2 \\ \vdots \quad \vdots & \vdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{np}x_p & = b_n \end{cases}$$

où les coefficients a_{ij} et b_i sont des réels fixés.

- ◊ Les inconnues sont x_1, x_2, ..., x_p.
- ◊ les réels aij sont les coefficients du premier membre. Le premier indice indique la ligne, le deuxième la colonne.
- ◊ Les réels b_i sont les coefficients du second membre. L’indice indique la ligne.

Ce projet porte sur les méthodes de résolution des systèmes linéaires soit celles directes(Gauss/Factorisation LU)ou celles itératives(Jacobi/Gauss Seidel).

2)-Déroulement de projet:

Le projet se déroule en deux phases principales:

- ➢ La Première phase consiste en la programmation basique du programme principal et des fonctions qui permettent la résolution du système Linéaire.
- ➢ La deuxième phase consiste en l’amélioration de ce programme Ces améliorations sont multiples et vous seront proposées dans un ordre de difficulté croissant à partir du moment ou la première phase est réalisée.

3)-Environnement de travail

- ✓ ***Environnment Matériel***: Ordinateur Portable avec un microprocesseur Intel Dual Core.
- ✓ ***Environnement Logiciel***: Windows 10 Pro
- ✓ ***Environnement de Développement***: Matlab R2021a

II. Objectif du projet:

Le projet a été conçu pour réaliser une interface qui va nous permettre de résoudre les systèmes linèare ainsi que minimizer le nombre de calcul par la réalisation d'une interface qui va afficher le résultat numérique par quatre méthodes différentes et le présenter graphiquement (2D).

Objectifs immédiats sont:

→ Faciliter la prise en main de Matlab.
→ Présenter les fonctions graphiques 2D.

III. Simulation:

Ci-dessous on va vous présenter les étapes suivis pour finaliser la réalisation de l'interface et les algorithmes utilisés.

D'abord nous allons définir une notion très importante qui est cnsidérée comme la base pour réaliser notre projet qui est la notion de *GUI/Guide *

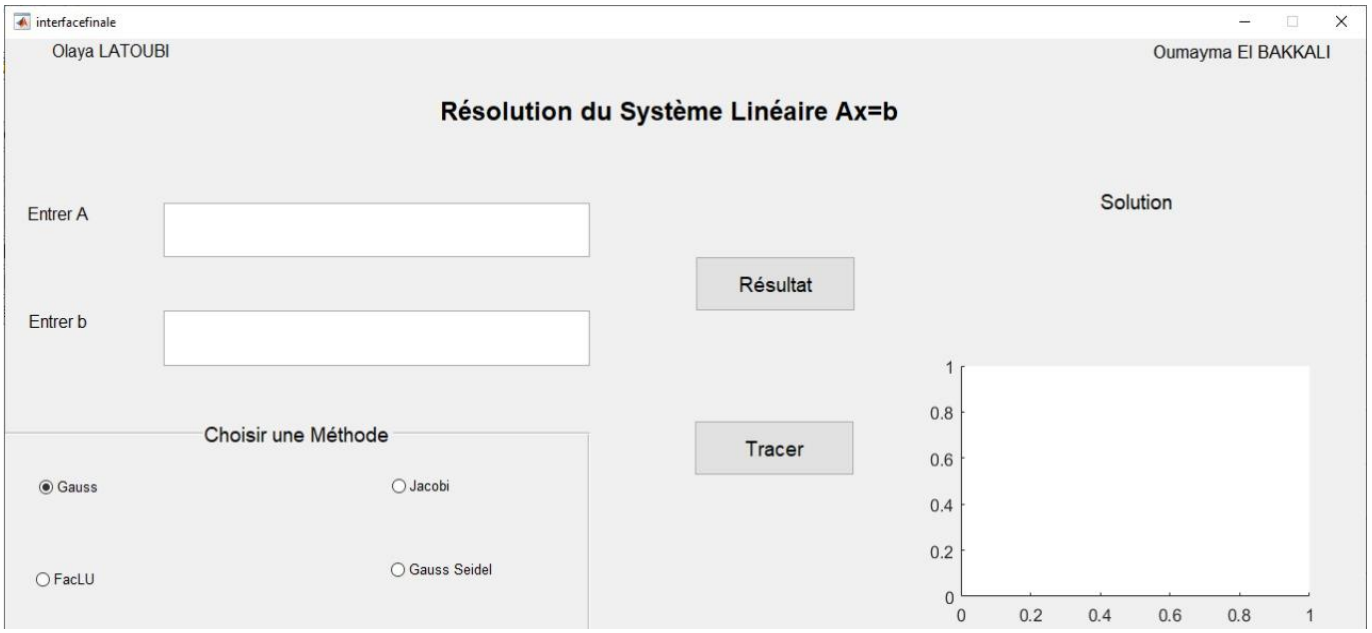
Le GUI (Graphical user Interface): permet à l'utilisateur d'interagir avec un programme informatique, grâce à différents objets graphiques .

Le GUI génère deux fichiers: **un fichier.fig** (non éditable) contenant les objets graphiques Figure, Axes et Pushbutton.

un fichier.m contenant le code du fonctionnement de interface Graphiques.

1)- Présentation du fichier.fig:

- Le fichier.fig du projet est présenté comme ci-dessous:



Cette interface est réalisée par L'utilisation d'un ensemble des objets graphiques; chacun de ces objets est associé à une fonction (callback) qui présente son fonctionnement.

N.B: Objets graphiques permettent, d'une part d'afficher les résultats de calculs sous des formes variées (point, courbe, surface, graphique) et d'autre part, de créer des interfaces graphiques (GUI) permettant à l'utilisateur d'interagir avec le programme.

2)-Présentation des objets grahiques:

- ✓ **Edit Text 1:** Espace pour entrer la matrice A

→ Matrice A doit obligatoirement être inversible.

- ✓ **Edit Text 2:** Espace pour entrer le vecteur b

- ✓ **Static Text 1:** Espace où la solution numérique doit être afficher.

- ✓ **Axes6:** Espace où la solution graphique doit être afficher(2D).

- ✓ **Uibuttongroup1:** Contient quatre radio-button nommés comme suit:

→ **radio-button1:** Gauss.

Voilà son callback:

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

% Hint: get(hObject,'Value') returns toggle state of gauss
global vall a b;
vall= get(handles.gauss,'value')
if vall==1
set(handles.facLU,'value',0)
set(handles.jacobi,'value',0)
set(handles. gaussseaidel,'value',0)
a = str2num(get(handles.matA,'string'));
b = str2num(get(handles.vecb,'string'));
end
```

→ **radio-buttoon2:** FacLU.

Voilà son Callback:

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

% Hint: get(hObject,'Value') returns toggle state of facLU
global val2 a b;
val2=get(handles.facLU,'value')
if val2==1
set(handles.gauss,'value',0)
set(handles.jacobi,'value',0)
set(handles. gaussseaidel,'value',0)
a = str2num(get(handles.matA,'string'));
b = str2num(get(handles.vecb,'string'));
end
```

→ **radio-buttoon3:** **J**acobi

Voilà son Callback:

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

% Hint: get(hObject,'Value') returns toggle state of jacobi
global val3 a b;
val3=get(handles.jacobi,'value')
if val3==1
set(handles.gauss,'value',0)
set(handles.facLU,'value',0)
set(handles. gaussseaidel,'value',0)
a = str2num(get(handles.matA,'string'));
b = str2num(get(handles.vecb,'string'));
end
```

→ **radio-buttoon4:** Gauss-Seidel.

Voilà son Callback:

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

% Hint: get(hObject,'Value') returns toggle state of gaussseaidelval1=get(handles.gauss,'value')
global val4 a b;
val4= get(handles.gaussseaidel,'value')
if val4==1
set(handles.gauss,'value',0)
set(handles. facLU,'value',0)
set(handles.jacobi,'value',0)
a = str2num(get(handles.matA,'string'));
b = str2num(get(handles.vecb,'string'));
end
```

Ce **uibuttongroup** fonctionne d'une manière booléenne. En effet si l'un des radio-button prend la valeur 1 les autres doivent prendre la valeur 0.

✓ **Push-button1:** Résultat.

-Explication-

Ce button affiche le résultat numérique de système linéaire par la méthode choisit par l'ustilisateur.

Le callback de ce button(résultat) contient les algorithmes des quatres méthodes Lorsque l'utilisateur choisit une parmi les quatre sa valeur dans uibuttongroup va prendre la valeur 1 et le système va être résolu selon cette dernière.

Ci-dessous on va vous afficher le Callback du button (Résultat) qui contient les algorithmes des 4 méthodes.

☺ **Avant de commencer nous allons faire un rappel sur chaque méthode**:

1ère partie du Callback: Méthode de Gauss

La méthode du pivot de Gauss est une méthode pour transformer un système en un autre système équivalent (ayant les mêmes solutions) qui est triangulaire et est donc facile à résoudre. Les opérations autorisées pour transformer ce système sont :

a. échange de deux lignes.
b. multiplication d'une ligne par un nombre non nul.
c. addition d'un multiple d'une ligne à une autre ligne.

Voilà Son code sous Matlab:

```
if val1==1
    A=[a b];
n=size(A,1);
for k=1:n-1
[C,I]=max(abs(A(k:n,k)));
if(C~=abs(A(k,k)))
A([(k+I-1) k],:)=A([k (k+I-1)],:);
end
for i=k+1:n
w=A(i,k)/A(k,k);
for j=k:n+1
A(i,j)=A(i,j)-w*A(k,j);
end
end
end
A(:,1:end-1)

A(:,end)
for i=n:-1:1
s=0;
for j=i+1:n
s=s+A(i,j)*x(j);
end
x(i)=(A(i,n+1)-s)/A(i,i);
end
```

★ **Principe:** (même principe dans le reste des méthodes)

Si l'utilisateur choisit Gauss comme la méthode de résolution sa valeur (val1) doit égale à 1 et les autres valeurs (val2, val3, val4) doivent êtes nulles.

2ère partie du Callback: Méthode de FacLU

Cette méthode permet de transformer une matrice carré A en un produit d'une matrice triangulaire inférieur L et d'une matrice triangulaire supérieur U. Cette décomposition permet notamment de résoudre des problèmes d'algèbre linéaire du type:
Ax=b <=> LUx=b

Voilà Son code sous Matlab:

```
elseif val2==1
     [L U]=lu(a);
x= U\(L\b);
```

3ère partie du Callback: Méthode de Jacobi

La méthode de Jacobi est une méthode itérative pour résoudre les systèmes linéaires Ax=b, où A est une matrice carrée d'ordre n et x, b sont des vecteurs de Rn. Elle consiste en la manipulation suivante : on décompose A comme A=D−E−F, où D est une matrice diagonale, −E est une matrice triangulaire inférieure, et −F est une matrice triangulaire supérieure.

Voilà Son code sous Matlab:

```
elseif val3==1
     x0=[0 ; 0 ; 0] ;
D=diag(diag(a));
L=tril(a,-1);
U=triu(a,1) ;
x1=1;
xi=x0 ;
while abs(x1-xi)>10^-6
     x1=-inv(D)*(L+U)*x0+inv(D)*b ;
     xi=x0 ;
     x0=x1 ;
end
x=x1;
```

Dernière partie du Callback: Méthode de Gauss-Seidel

La méthode de Gauss-Seidel est une méthode pour résoudre les systèmes linéaires Ax=b, où A est une matrice n×n et x,b sont des vecteurs de $\mathbf{R}^n$. Elle consiste en la manipulation suivante : on décompose A comme A=D-E-F, où D est une matrice diagonale, -E est une matrice triangulaire inférieure, et -F est une matrice triangulaire supérieure.

Voilà Son code sous Matlab:

```
else
     x0=[0 ; 0 ; 0] ;
D=diag(diag(a));
L=tril(a,-1);
U=triu(a,1) ;
x1=1;
xi=x0 ;
while abs(x1-xi)>10^-6
     x1=-inv(D+L)*U*x0+inv(D+L)*b ;
     xi=x0 ;
     x0=x1 ;
end
x=x1 ;
```

- ✓ **Push-button2:** Tracer.

Ce button nous permet de tracer la solution graphique(2D) sous forme de deux droite leurs intersections (un point) est la solution graphique.

Voilà son Callback:

```
% --- Executes on button press in draw.
function draw_Callback(hObject, eventdata, handles)
% hObject    handle to draw (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global a b ;
[n,m] =size(a);
if n==2
    x= -5:0.1:5;
    x1 = (b(1,1)/a(1,1))-((a(1,2)*x)/a(1,1));
    x2= (b(2,1)/a(2,1))-((a(2,2)*x)/a(2,1));
    axes(handles.axes6);
    plot(x,x1,'r');
    hold on;
    plot(x,x2,'k');
    hold off;

end
```

IV. Conclusion:

Ce projet s'est révélé très enrichissant dans la mesure où il a consisté en une approche concrète du métier d'ingénieur. En effet, pendant la réalisation de ce projet on a rencontré quelques difficultés (parmi ces eux: manque de sources d'information.) Mais cela nous a poussé à faire plus d'effort pour atteindre notre objectif avec le respect total d'un délais de 4 semaines.

Pour la première fois, nous avons mené à réaliser un tel projet qui nous a aider à savoir comment ustiliser les GUI sous Matlab ainsi que les fonctions graphiques(2D).

Enfin nous devons avouer que nous sommes satifaites de ce projet puisque nous avons atteint des nouveaux objectifs. Nous sommes par ailleurs convaincus que le travail élaboré n'est qu'une étape primaire ausi bien pour une carrière proffessionnelle que pour des études plus approfondies.

V. Bibliographie:

MATLAB - MathWorks: https://ch.mathworks.com/products/matlab.html

Youtube-Channel: https://www.youtube.com/watch?v=wxiwcmMW1C8&t=134s