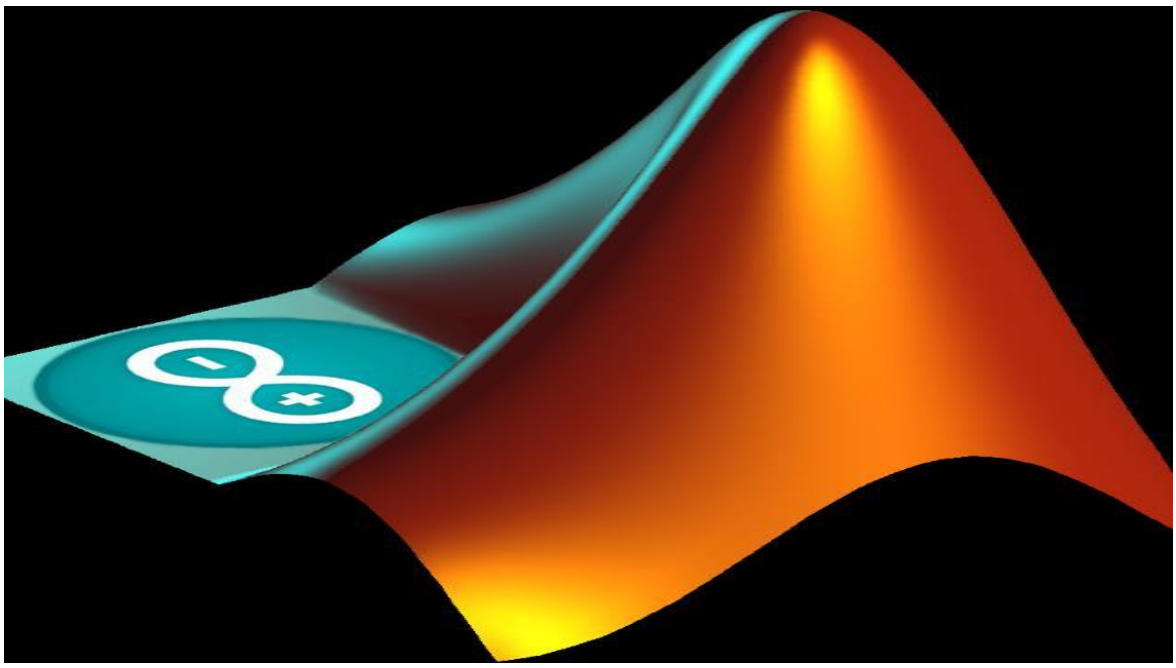
Université EUROMED de Fès
Ecole d'Ingénierie Digitale et
d'Intelligence Artificielle

Résolution d'un système linéaire Ax=b



Encadré Par :
Dr. Meryem ELMOUHTADI

Réaliser par :
-Ayoub Hsaine
-Mohamed Amine Kebdani

Groupe : D1D2E1

Année universitaire 2020-2021

Sommaire :

Remerciement

On remercie dieu le tout puissant de nous avoir donné la santé et la Volonté d'entamer et de terminer ce projet.

Je tiens à remercier dans un premier temps, toute l'équipe pédagogique de l'école d'ingénierie digitale et d'intelligence artificielle.

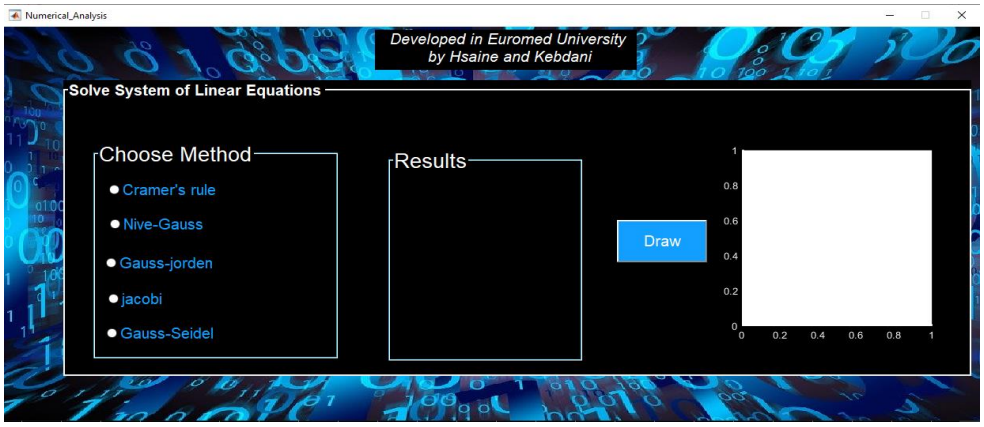
Avant, d'entamer ce rapport nous profitons de l'occasion pour remercier tout d'abord notre professeur Dr. Meryem ELMOUHTADI qui n'a pas cessé de nous encourager pendant la durée du projet, ainsi pour sa générosité en matière de formation et d'encadrement, pour la qualité de son encadrement exceptionnel, pour sa patience, sa rigueur et sa disponibilité durant notre préparation de ce projet. Nous le remercions également pour l'aide et les conseils concernant les missions évoquées dans ce rapport, qu'il nous a apporté lors des différents suivis, et la confiance qu'il nous a témoigné.

Introduction

Les interfaces graphiques (ou interfaces homme-machine) sont appelées GUI (pour Graphical User Interface) sous MATLAB. Elles permettent à l'utilisateur d'interagir avec un programme informatique, grâce à différents objets graphiques (boutons, menus, cases à cocher…). Ces objets sont généralement actionnés à l'aide de la souris ou du clavier.Dans le cadre de la deuxième année de cycle préparatoire nous avons réalisé une interface graphique permettant à l'utilisateur de résoudre un système linéaire Ax=b.

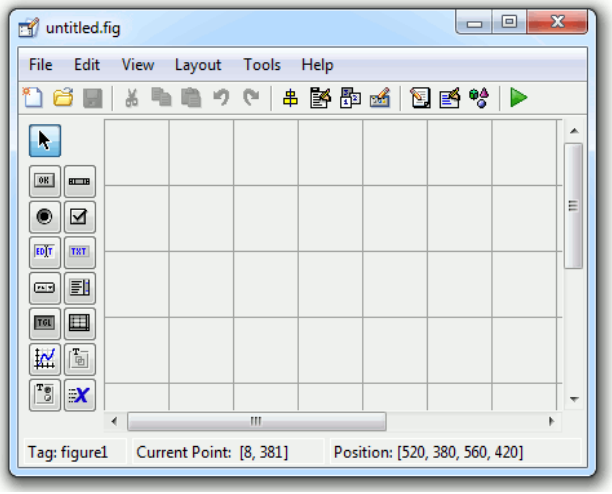
Grâce à ce projet nous avons eu l'opportunité de cumuler les connaissances théoriques (Cours Analyse numérique) avec celle de la pratique (Matlab), ceci permet également de rentrer dans la vie active et découvrir plus précisément le milieu professionnel.

Ce projet consiste à découvrir le 'GUIDE' de logiciel Matlab, celui-là facilite la tâche de calcul d'un système linéaire, ce calcul peut se faire en différents méthodes selon le choix de l'utilisateur mais tous les méthodes ramène au même résultat, la seule différence entre ces méthodes est le cout d'exécution, cette interface affiche le résultat numériquement puis graphiquement en appuyions sur le buton DRAW.



Partie I : Présentation de l'outil GUIDE

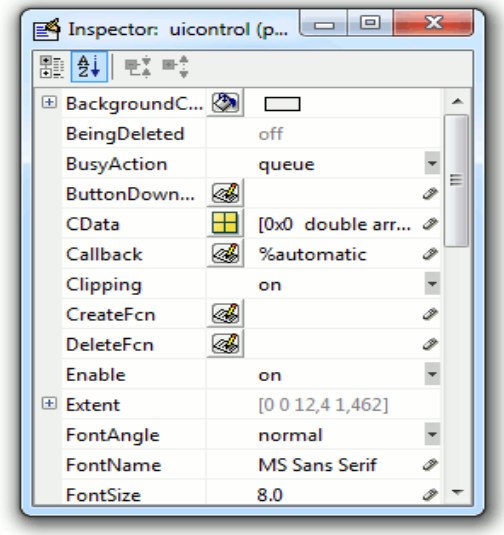
- **L'outil guide**



Le guide est un constructeur d'interface graphique qui regroupe tous les outils dont le programmeur a besoin pour créer une interface graphique de façon intuitive. Il s'ouvre, soit en cliquant sur l'icône, soit en tapant guide dans le Command Window de Matlab.

Le placement des objets est réalisé par sélection dans une boite à outils. Leur mise en page et leur dimensionnement se font à l'aide de la souris.

- **Modification des propriétés d'un Button :**



Un double-clique sur un objet permet de faire apparaitre le Property Inspector ou les propriétés des objets sont facilement éditables Leurs modifications et la visualisation de ces modifications sont immédiates.

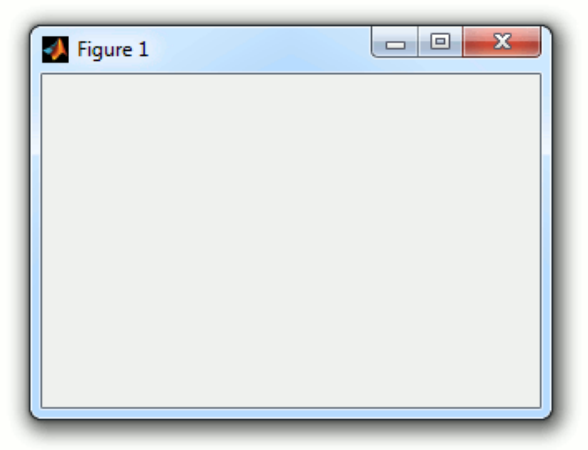
- **Enregistrement :**

Une fois l'interface graphique terminée, son enregistrement donne deux fichiers portant le même nom mais dont les deux extension sont.fig et.m.

Le fichier.fig contient la définition des objets graphiques.

Le fichier.m contient les lignes de code qui assurent le fonctionnement de l'interface.

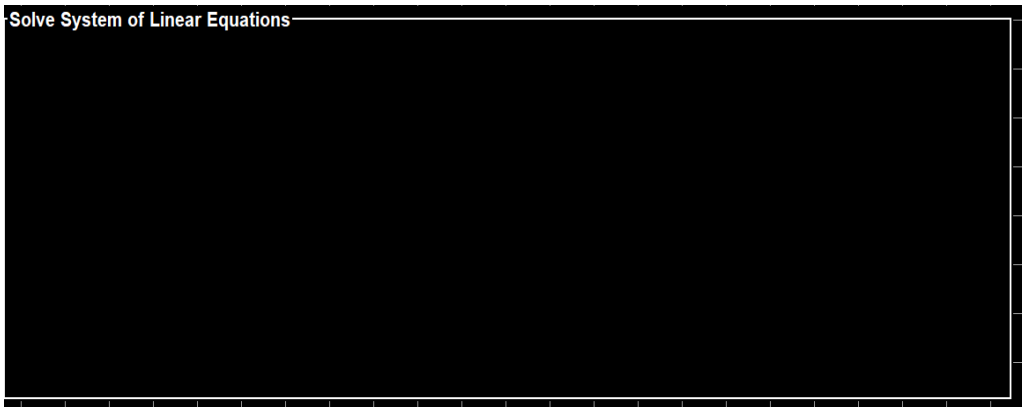
1. Objets figure



Les objets figure sont les conteneurs ou sont disposés tous les autres objets enfants.

2. Objets UI

2.1. Panel



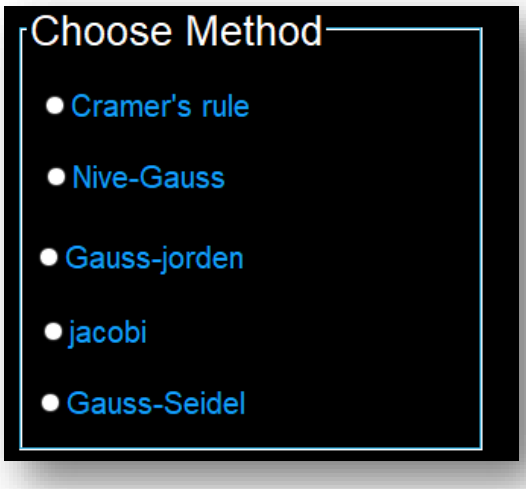
Dans le panel nous avons regroupé toutes les autres composantes

2.2. Texte

Developed in Euromed University
by Hsaine and Kebdani

Cet Button est de type statique qui permet d'écrire un texte quelconque comme le montre l'exemple.

2.3.Button group et radio Button

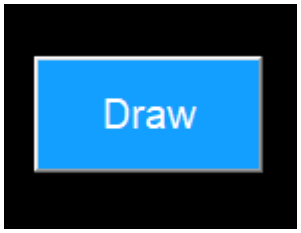


Button group est un Button qui regroupe un ensemble des radio Button, celui-là permet à l'utilisateur de choisir une méthode parmi les options existantes pour résoudre le système.

2.4.Panel résultat

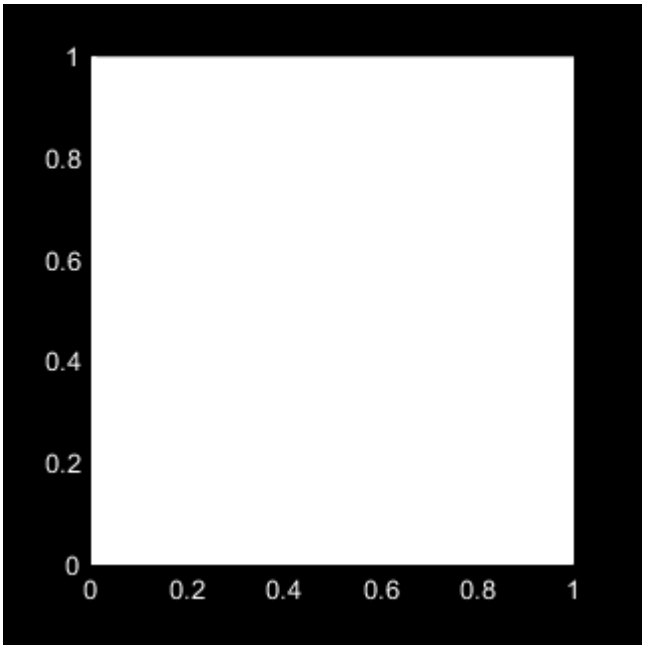
Ce panel possède à l'intérieur un Button de type texte qui affiche le résultat numérique du système linéaire.

2.5.Push Button



Cette Button est de type push Button nommé DRAW permet d'afficher le résultat graphique quand vous le cliqué, elle est liée avec panel résultat et axes12.

3. Objets Axes



L'objet axes12 est une zone de traçage des graphiques

4. Callback

Callback permet d'accéder à fichier.m(Numerical_Analysis.m) à partir du fichier.fig(Numerical_Analysis.ig).

Afin de connecter les buttons, on doit mettre le code de chaque Button dans sa partie du fichier.fig correspondante,en suivant ces etapes :

clique droite (sur le button) -> view callbacks -> callback

Partie II : Code des Objets

1. Background

Pour implémenter une image dans le background de l'interface nous utilisons le code suivant :

```
ha = axes('units','normalized', ...
            'position',[0 0 1 1]);
% Move the background axes to the bottor
uistack(ha,'bottom');
% Load in a background image and displa
% The image used below, is in the Image
I=imread('images.jpg');
hi = imagesc(I)
colormap gray
% Turn the handlevisibility off so that
% Also, make the axes invisible
set(ha,'handlevisibility','off', ...
            'visible','off')
axes('position',[0.3,0.35,0.4,0.4])
plot(rand(10))
```

ha=axes() :pour créer les axes du background

Imread : pour lire l'image

La commande set : pour rendre les axes invisibles

2. Commande commune

Handles.tag lire objet (radio button pop menu…)

L'instruction get obtient la valeur de chaine à partir d'un composant d'entré. Par Example si nous voulons obtenir le numéro saisi cramer, nous pouvons le faire comme ceci (voir l'implémentation dans toutes les méthodes utilisées ci-dessous

```
get(handles.edit1, 'String' )
```

L'instruction set définit les priorités de l'élément que vous indiquez la priorité Tag est l'identifiant à utiliser à cet effet.

```
set(handles.output_line, 'String' ,
```

Prompt1 :affiche la case approprié pour entrer la matrice et le vecteur.

```
                                    Enter a column vector of constants '};
dlgtitle1 = 'Inputs';
dims1 = [1 35];
definput1 = {'',''};
answer5 = inputdlg(prompt1,dlgtitle1,dims1,definput1);
A=str2num(answer5{1});
b=str2num(answer5{2});
```

3. Algorithme :

3.1. Cramer

```
function cramer_Callback(hObject, eventdata, handles)
val=get(handles.cramer,'value');
if val==1
set(handles.ssos, 'String','')
set(handles.ng,'value',0)
set(handles.gj,'value',0)
set(handles.jac,'value',0)
set(handles.gs,'value',0)
prompt1 = {'Enter the matrix of coefficients ','Enter a column vector of constants '};
dlgtitle1 = 'Inputs';
dims1 = [1 35];
definput1 = {'',''};
answer5 = inputdlg(prompt1,dlgtitle1,dims1,definput1);
A=str2num(answer5{1});
b=str2num(answer5{2});
```

```
n=length(b);
res=zeros(n,1);
D=det(A);
for i=1:n
    Aug=A;
    Aug(:,i)=b;
    res(i)=(det(Aug)/D);
end
x=res;
x=x(:);
nn=length(x);
 for xcm = 1:nn
          old_str=get(handles.ssos, 'String');
          tmp_str=['X',num2str(xcm),' = ',num2str(x(xcm))];
          new_str=[old_str; {tmp_str}];
          set(handles.ssos, 'String', new_str);
 end
end
```

3.2. Naive-gauss

```
val=get(handles.ng,'value');
if val==1
set(handles.ssos, 'String','')

set(handles.cramer,'value',0)
set(handles.gj,'value',0)
set(handles.jac,'value',0)
set(handles.gs,'value',0)
prompt1 = {'Enter the matrix of coefficients ','Enter a column vector of
dlgtitle1 = 'Inputs';
dims1 = [1 35];
definput1 = {'',''};
answer5 = inputdlg(prompt1,dlgtitle1,dims1,definput1);

a=str2num(answer5{1});

b=str2num(answer5{2});

ab = [a,b]
[R, C] = size(ab);
for j = 1:R-1
% Pivoting section starts
    if ab(j,j)==0
        for k=j+1:R
            if ab(k,j)~=0
                abTemp=ab(j,:);
                ab(j,:)=ab(k,:);
                ab(k,:)=abTemp;
                break
            end
        end
    end
% Pivoting section ends
    for i = j+1:R
        ab(i,j:C) = ab(i,j:C)-ab(i,j)/ab(j,j)*ab(j,j:C);
    end
end
x = zeros(R,1);
x(R) = ab(R,C)/ab(R,R);
for i = R-1:-1:1
    x(i)=(ab(i,C)-ab(i,i+1:R)*x(i+1:R))/ab(i,i);
end


x=x(:);
nn=length(x);
 for xcm = 1:nn
          old_str=get(handles.ssos, 'String');
          tmp_str=['X',num2str(xcm),' = ',num2str(x(xcm))];
          new_str=[old_str; {tmp_str}];
          set(handles.ssos, 'String', new_str);
 end
```

3.3. Gauss Jorden

```
a=str2num(answer5{1});

b=str2num(answer5{2});
ab = [a,b];
[R, C] = size(ab);
for j = 1:R
    % Pivoting section starts
    pvtemp=ab(j,j);
    kpvt=j;
```

```
for j = 1:R
    % Pivoting section starts
    pvtemp=ab(j,j);
    kpvt=j;
 % Looking for the row with the largest pivot element.
    for k=j+1:R
        if ab(k,j)~=0 && abs(ab(k,j)) > abs(pvtemp)
          pvtemp=ab(k,j);
          kpvt=k;
        end
    end
 % If a row with a larger pivot element exists, switch the rows.
    if kpvt~=j
       abTemp=ab(j,:);
       ab(j,:)=ab(kpvt,:);
       ab(kpvt,:)=abTemp;
    end
 % Pivoting section ends
    ab(j,:)= ab(j,:)/ab(j,j);
    for i = 1:R
        if i~=j
           ab(i,j:C) = ab(i,j:C)-ab(i,j)*ab(j,j:C);
        end
    end
end
```

```
x=ab(:,C);
x=x(:);
x=x(:);
nn=length(x);
 for xcm = 1:nn
          old_str=get(handles.ssos, 'String');
          tmp_str=['X',num2str(xcm),' = ',num2str(x(xcm))];
          new_str=[old_str; {tmp_str}];
          set(handles.ssos, 'String', new_str);
 end

end
```

3.4. Jacobi

```
val=get(handles.jac,'value');
if val==1
set(handles.ssos, 'String','')
set(handles.cramer,'value',0)
set(handles.gj,'value',0)
set(handles.ng,'value',0)
set(handles.gs,'value',0)
prompt1 = {'Enter the matrix of coefficients ','Enter a column vector of constants ','Enter initial guess fo
dlgtitle1 = 'Inputs';
dims1 = [1 35];
definput1 = {'','',''};
answer5 = inputdlg(prompt1,dlgtitle1,dims1,definput1);

A=str2num(answer5{1});

b=str2num(answer5{2});
x=str2num(answer5{3});


n=size(x,1);
normVal=Inf;
%%
% *  *Tolerence for method*
```

```
while normVal>tol
    xold=x;
    for i=1:n
        sigma=0;
        for j=1:n
            if j~=i
                sigma=sigma+A(i,j)*x(j);
            end
        end
        x(i)=(1/A(i,i))*(b(i)-sigma);
    end
    itr=itr+1;
    normVal=abs(xold-x);
end
x=x(:);
nn=length(x);
 for xcm = 1:nn
          old_str=get(handles.ssos, 'String');
          tmp_str=['X',num2str(xcm),' = ',num2str(x(xcm))];
          new_str=[old_str; {tmp_str}];
          set(handles.ssos, 'String', new_str);
 end
end
```

3.5. Gauss Seidel

```
% Hint: get(hObject,'Value') returns toggle state of gs
val=get(handles.gs,'value');
if val==1
set(handles.ssos, 'String','')
set(handles.cramer,'value',0)
set(handles.gj,'value',0)
set(handles.ng,'value',0)
set(handles.jac,'value',0)
prompt1 = {'Enter the matrix of coefficients ','Enter a column vector of constants ','Enter initial guess for the
dlgtitle1 = 'Inputs';
dims1 = [1 35];
definput1 = {'','',''};
answer5 = inputdlg(prompt1,dlgtitle1,dims1,definput1);

A=str2num(answer5{1});

b=str2num(answer5{2});

x=str2num(answer5{3});

n=size(x,1);
normVal=Inf;
%%
% * _*Tolerence for method*_
tol=1e-5; itr=0;
```

```
%% Algorithm: Gauss Seidel Method
while normVal>tol
    x_old=x;
    for i=1:n
        sigma=0;
        for j=1:i-1
                sigma=sigma+A(i,j)*x(j);
        end
        for j=i+1:n
                sigma=sigma+A(i,j)*x_old(j);
        end
        x(i)=(1/A(i,i))*(b(i)-sigma);
    end
    itr=itr+1;
    normVal=norm(x_old-x);
end
x=x(:);
nn=length(x);
 for xcm = 1:nn
          old_str=get(handles.ssos, 'String');
          tmp_str=['X',num2str(xcm),' = ',num2str(x(xcm))];
          new_str=[old_str; {tmp_str}];
          set(handles.ssos, 'String', new_str);
 end
end
```

4. Push Button Draw

```
function pushbutton9_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton9 (see GCBO)
xi = [-2 -1.5 -1 0 1 1.5 2];
yi = [3.5  2 1.5 0 1.3 2.5 3.9];

A3 = polyfit (xi, yi, 3)
 x=-2:0.1:2
plot(xi, yi, 'o', x, polyval(A3,x), '--' );
% handles    structure with handles and user data (see GUIDATA)
```

Notre représentation graphique est sous forme d'une approximation linéaire par la commande polyfit, cette méthode renvoie la qualité de l'approximation réalisé

Conclusion

En conclusion, nous devons avouer que rétrospectivement nous sommes satisfaits de ce projet puisque nous avons atteint des nouveaux objectifs et des nouvelles technologies.

En effet ce projet nous a permis de comprendre les outils existants dans le logiciel Matlab et apprendre comment réaliser une interface graphique GUI sous Matlab.

L'utilisation de 'guide' de Matlab à faciliter la tâche de la réalisation de l'interface, nous avons appris comme des nouvelles techniques :

- La construction des différentes composantes comme pop menu, radio Button texte statique, panel ... Leur fonctionnement, jouer sur la dimension les couleurs les bordures ...
- la manière de connecter tous les composants au script de commande grâce à'' view->callback'' qui simule la connexion, ainsi comment gérer nos temps, stress et surtout nos projets.

Enfin nous ne prétendons pas avoir résoudre le problème pose dans son intégrité mais nous sommes par ailleurs convaincus que le travail élaboré n'est qu'une étape primaire aussi bien pour une carrière professionnelle que pour des études plus approfondies.

Bibliographie

Concernant la bibliographie de ce projet, nos recherches ont été principalement concentré sur le site principal de Matlab « MathWorks » dont le lien est https://fr.mathworks.com/, et les liens suivants :

https://www.tutorialspoint.com/matlab/index. html

https://www.matrixlab-examples.com/callback-function.html

https://nte.mines-albi.fr/MATLAB/co/cnApproximationLineaire.html