UPEC
UNIVERSITÉ
PARIS-EST CRÉTEIL
VAL DE MARNE
Connaissance - Action

EPISEN
École Publique d'Ingénieurs de la Santé Et du Numérique

Conception d'une interface

Projet Matlab

Encadrant : M. FOURNIER

Projet réalisé par :
L'équipe Violette : Manal NEJMI - Alicia FERREIRA - Marina KHALIFA

ITS 2 - Semestre 3

Année universitaire : 2021 - 2022

Remerciements

Avant de commencer, nous tenons à remercier toutes les personnes qui ont contribué à l'accomplissement de ce projet.

Nous adressons nos remerciements dans un premier temps à M. FOURNIER qui nous a accompagné tout au long de ce projet, pour son aide et ses conseils dans plusieurs étapes du projet. Grâce aux cours qu'il nous a dispensé, on a pu développer nos compétences en Matlab.

Nous voulions également remercier Mr MELLOUK Abdelhamid, le directeur de la filière ITS de l'EPISEN sans qui tout ceci n'aurait été possible.

Table des matières

Liste des figures

Introduction

Au cours du module Systèmes des mesures nomades, nous avions un projet de TP à réaliser avec notre enseignant Mr FOURNIER. Celui-ci a pour but de développer une interface homme-machine sous un outil de programmation : Matlab. Cette interface va permettre de traiter une fonction qui simule un signal, d'appliquer des erreurs et de pouvoir éventuellement les corriger. En conséquence, on doit visuellement être capable de constater ces erreurs.

Cette interface comportera différents curseurs de données qui seront instancier par l'utilisateur et généra différents graphiques.

Afin de réaliser ce projet de TP, nous avons travaillé en groupe. Notre équipe est constituée de KHALIFA Marina, NEJMI Manal et FERREIRA Alicia.

Ce projet nous a permis de mettre en œuvre nos compétences en programmation, notamment le codage en Matlab pour générer des interfaces homme-machine, appliquer des fonctions mathématiques et en gestion de projet.

Objectifs

Les objectifs principaux de ce projet sont de :

- Simuler un signal harmonique
- Imposer différentes erreurs au signal simulé
- Être en mesure de les corriger

Notions théoriques

- **Les capteurs :**

Le capteur va transformer une grandeur physique en une grandeur normée (souvent électrique) pouvant être interprétée par un dispositif de commande / contrôle.

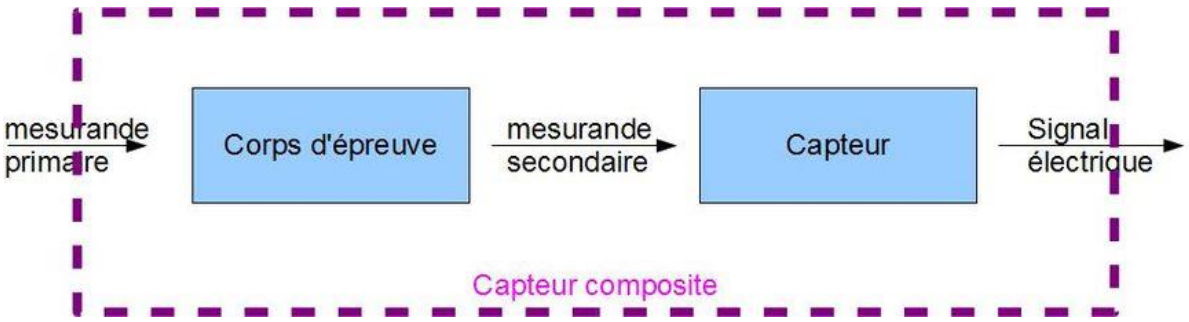

Figure 1 : Principe d'un capteur

- **Les caractéristiques d'un capteur :**

Les capteurs peuvent être caractérisés par 6 caractéristiques :

- Etendue de mesure : Les valeurs extrêmes pouvant être mesurées par le capteur.
- Résolution : Plus petite variation de grandeur qui sera mesurable.
- Sensibilité : La variation du signal de sortie par rapport à la variation du signal d'entrée.
- Rapidité : Temps de réaction du capteur.
- Linéarité : Ecart de sensibilité sur l'étendue de mesure.

- **Les erreurs possibles d'un capteur :**

Afin de concevoir un capteur, S doit dépendre le moins possible de :

- La valeur de m (linéarité)
- La fréquence de variation (bande passante)
- Du temps (vieillissement)
- D'actions extérieures (grandeur d'influence)

$$\Delta s = S.\ \Delta m$$

Parmi les erreurs les plus possibles et connues, on trouve :

- **Erreur de gain ou d'échelle :** correspond à l'application d'un certain coefficient à la vraie courbe. Elle est visible essentiellement sur la borne supérieure de l'étendue de mesure.

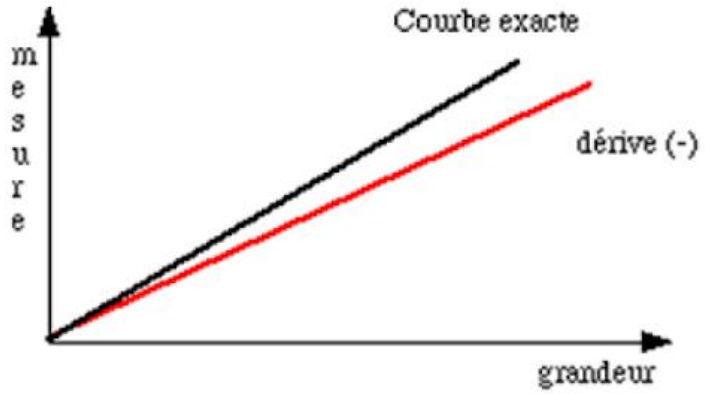

Figure 2 : Effet de l'erreur de gain ou d'échelle

- **Erreur d'hystérésis** : c'est l'écart maximum (rapporté à la valeur absolue) de la valeur du signal de sortie à la ligne droite de référence.

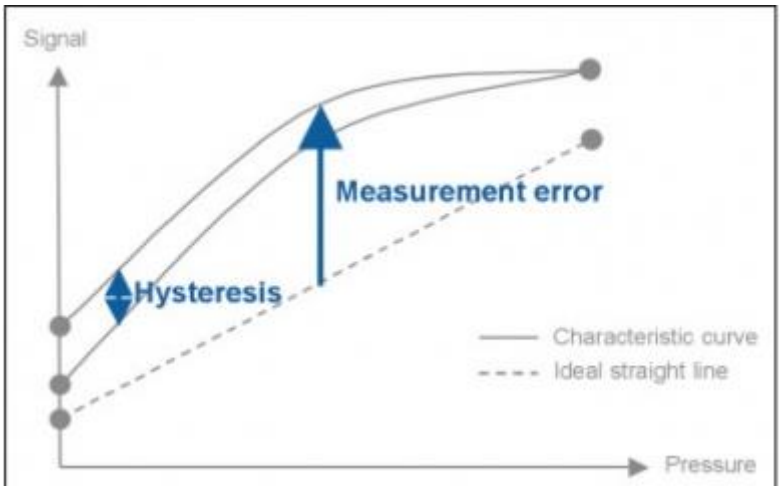

Figure 3 : Effet de l'erreur d'hystérésis

- **Erreur d'offset :** ou décalage, c'est la différence entre la valeur « vraie » de la mesure et celle obtenue à partir de la réponse du capteur pour la borne inférieure de l'étendue de mesure.

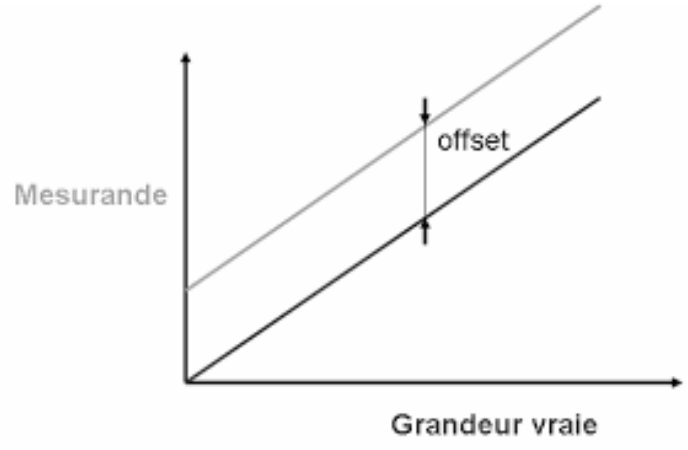

Figure 4 : Effet de l'erreur offset

- **Erreur de linéarité :** Valeur absolue de l'écart maximum entre la courbe caractéristique d'un capteur de couple déterminée, par valeurs montantes à la ligne droite de référence qui rapproche la courbe caractéristique d'une ligne droite idéale.

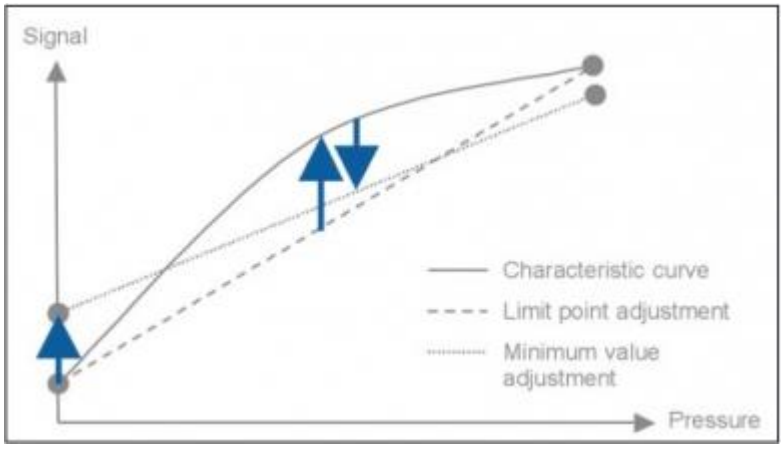

Figure 5 : Effet de linéarité

- **Erreur de non-linéarité :** L'erreur de non-linéarité représente le plus grand écart entre la courbe d'étalonnage et une ligne droite appelée "meilleure droite" reliant le signal de sortie à la force appliquée.

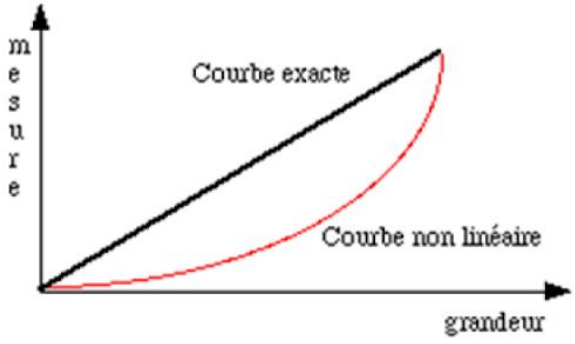

Figure 6 : Effet de non-linéarité

- **Grandeur d'influences**

La réponse d'un capteur peut être modifier selon plusieurs grandeurs :

- Température : mécaniques, dimensionnelle
- Pression, vibrations
- Humidité : Permet la variation des propriétés électriques (Constante diélectrique ou résistivité de l'isolation électrique)
- Champs magnétiques : par rapport aux champs variables d'induction ou des modifications électrique pour les variables et champs statiques
- Tension d'alimentation : problème d'amplitude ou de fréquence.

Modélisation : résolution sous Matlab

Au niveau de cette partie, nous allons modéliser les erreurs abordées précédemment avec notre programme sous Matlab. Cela a pour but de comprendre les effets de chaque erreur d'un capteur sur notre fonction et comment procéder à leur correction. On va également tester avec la fréquence d'échantillonnage pour calculer la transformée de Fourrier pour avoir un spectre exploitable.

Il faut savoir qu'il est important de récupérer fe et de vérifier qu'on a bien fe/2 car au-delà de fe/2, on ne pourra pas exploiter la courbe.

Nos entrées sont : A1, A2, A3, f1, f2, f3, fe, M, G, O

Nos sorties sont : se, t, y, Ynew, Ybis, vf, SE, d, j, h, e, q, r

- **A1/2/3 :** amplitudes
- **G :** gain
- **M :** moyenne
- **O :** offset
- **fe :** fréquence d'échantillonnage
- **se :** signal sinusoïdal auquel une erreur linéaire a été appliquée (gain et offset)
- **t :** vecteur allant de 0 à 1 avec un pas de Te
- **y :** signal sinusoïdal de sortie
- **Ynew :** transformé de Fourier du signal y sans erreur
- **Ybis :** transformé de Fourier du signal y avec erreur
- **vf** : vecteur fréquence allant de 0 à fe avec un pas de 1

- **SE :** transformée de Fourrier du Signal sinusoïdal auquel une erreur linéaire a été appliquée (gain et offset)
- **d :** polynôme appliquée à t (p)
- **j :** polynôme appliquée au signal de base y (z)
- **h :** exponentielle appliquée à t
- **e :** exponentielle appliquée au signal de base y
- **q :** logarithme appliquée à t (o)
- **r :** logarithme appliquée au signal de base y (l)

Nos fonctions de simulation sont :

Fonction de signal	Transformé de Fourrier	Non-linéarité
[se,t,y]=fonction_signal(M,A1,A2,A3,f1,f2,f3,fe,G,O)	[Ynew,Ybis,t,y,vf]=fonction_TF(M,A1,A2,A3,f1,f2,f3,fe,G,O)	[t,y,vf,Ynew,SE,se,d,j,h,e,q,r]= fonction_non_lineaire(M,A1,A2,A3,f1,f2,f3,fe,G,O,a,b,c,lmbd,lambda,mu,mu1)

- Fonction 1 : Signal harmonique et spectre

Dans un premier temps, nous avons codé notre première fonction de notre signal harmonique temporel.

y=M+A1*sin(2*pi*f1*t)+ A2*sin(2*pi*f2*t) + A3*sin(2*pi*f3*t).

Nous avons défini également un vecteur Temps : t=[0:Te:1] avec Te= 1/fe.

```
function [se,t,y]=fonction_signal(M,A1,A2,A3,f1,f2,f3,fe,G,O)

Te=1/fe
t=[0:Te:1];

y=M+A1*sin(2*pi*f1*t)+ A2*sin(2*pi*f2*t) + A3*sin(2*pi*f3*t);%fonction
```

Figure 7 : Fonction signal

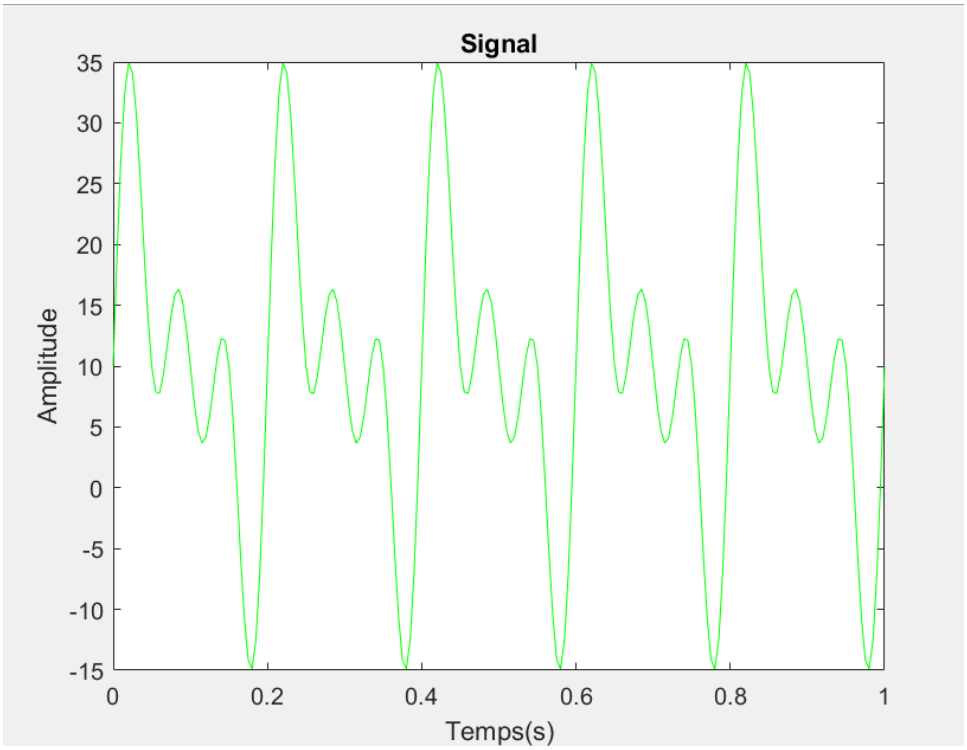

Figure 8 : Graphique du signal y sans erreur

Les valeurs de A1, A2, A3 vont donc être choisies par l'utilisateur à l'aide des sliders.

- **Fonction 2 : Fonction de la transformée de Fourrier**

Dans un second temps, nous avons traité une autre fonction avec la fréquence d'échantillonnage fe qui représente le nombre d'échantillons dans notre signal. Dans ce cas-là, nous avons modifier le pas dans notre vecteur temps par la période de l'échantillonnage 1/fe. Afin d'avoir la fréquence f1, f2, f3 plus élevée que fe, nous avons élargi ces valeurs de telle sorte que la valeur la plus grande de ces fréquences soit à la moitié de la fréquence d'échantillonnage.

En appliquant la transformée de Fourrier, nous avons mis une estimation à notre spectre via la fonction « ftt », en basculant du domaine temporel au domaine fréquentiel. Nous avons donc créé un vecteur fréquence allant de 0 à fe avec un pas de 1. Nous avons ajusté les amplitudes Y(1,1). Nous avons donc deux vecteurs : entre 0 et fe/2 c'est la partie positive de notre spectre, et entre fe/2 et fe c'est la partie négative de notre spectre.

Afin d'afficher que la partie positive du spectre, nous avons donc utilisé la fonction « floor », ce a pour but de respecter le théorème de Shanon. Cette fonction permet donc de ne plus avoir les valeurs négatives

```
function [Ynew,Ybis,t,y,vf]=fonction_TF(M,A1,A2,A3,f1,f2,f3,fe,G,O)
Te=1/fe
t=[0:Te:1];

y=M+A1*sin(2*pi*f1*t)+ A2*sin(2*pi*f2*t) + A3*sin(2*pi*f3*t);

%Transformé de fourrier
Y=abs(fft(y))/fe;

%GAIN ET OFFSET
%on ajoute le gain au niveau du graph 1 (superposition des deux courbes)
se= G * y + O;
% x=t*fe
vf=[0:1:fe];
Ynew=[Y(1,1), 2*Y(1,2:floor(fe/2))];%notre moyenne est Y(1,1) cette ligne pour ajuster tout les amplitudes à 10
% Ynew=[Y(1,1), 2*Y(1,2:end)]
SE=abs(fft(se))/fe;
vf=[0:1:fe];

Ybis=[SE(1,1), 2*SE(1,2:floor(fe/2))]
```

Figure 9: Fonction de transformée de Fourrier

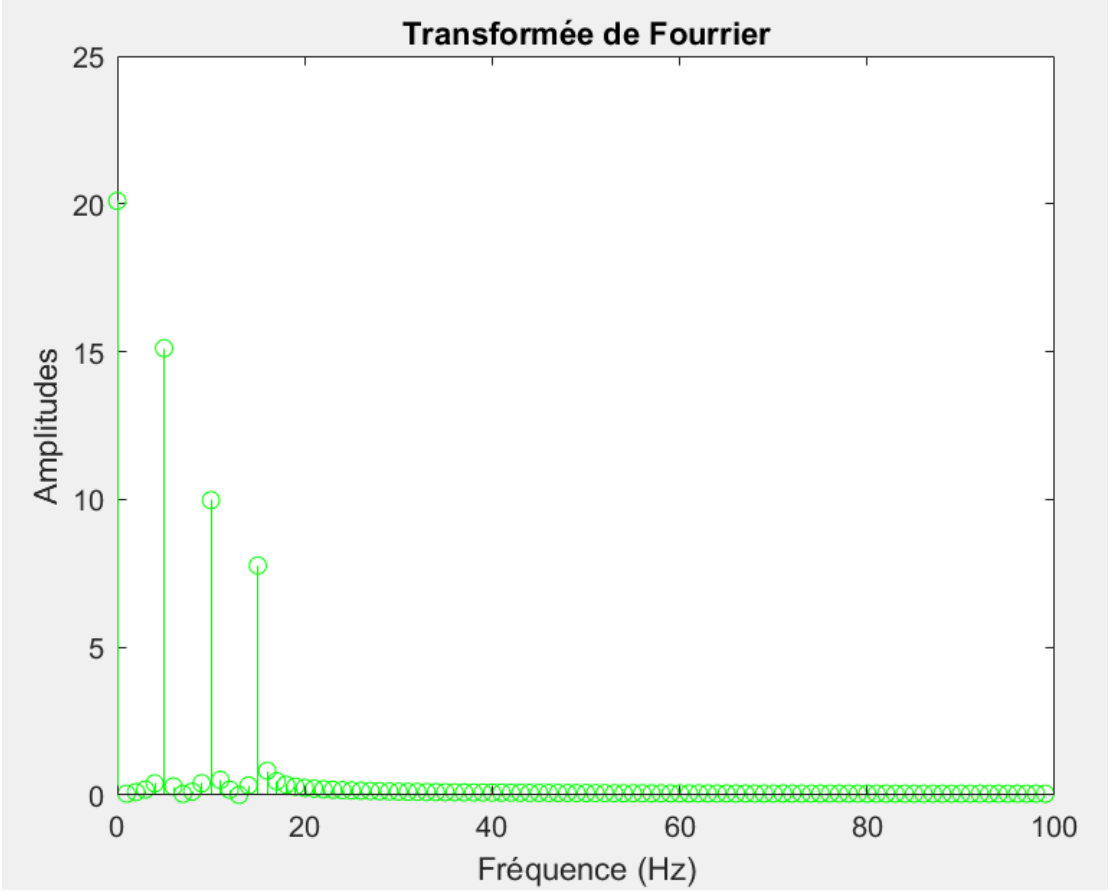

Figure 10 : Graphique de la transformée de Fourrier sans erreur

Nous avons donc conservé qu'une partie de notre signal, et multiplier la partie négative par 2 pour ne pas perdre le signal tout en gardant la même moyenne.

- **Gain**

Afin d'observer la différence entre les deux courbes avec la superposition, nous avons multiplier le signal par le Gain. Nous avons remarqué que la fréquence d'échantillonnage fe modifie les fréquences de nos amplitudes. Pour cela nous avons choisis d'agrandir nos amplitudes par rapport à la fréquence.

- **Offset**

On a rajouté à notre fonction de base une valeur moyenne. On a appliqué la transformée de Fourrier par rapport à notre signal qui est devenu SE. L'offset est donc les valeurs à soustraire de notre spectre de la valeur initiale.

```
% GAIN ET OFFSET
%on ajoute le gain au niveau du graph 1 (superposition des deux courbes)

se= G * y + O
```

Figure 11 : Fonction de signal auquel une erreur a été appliqué

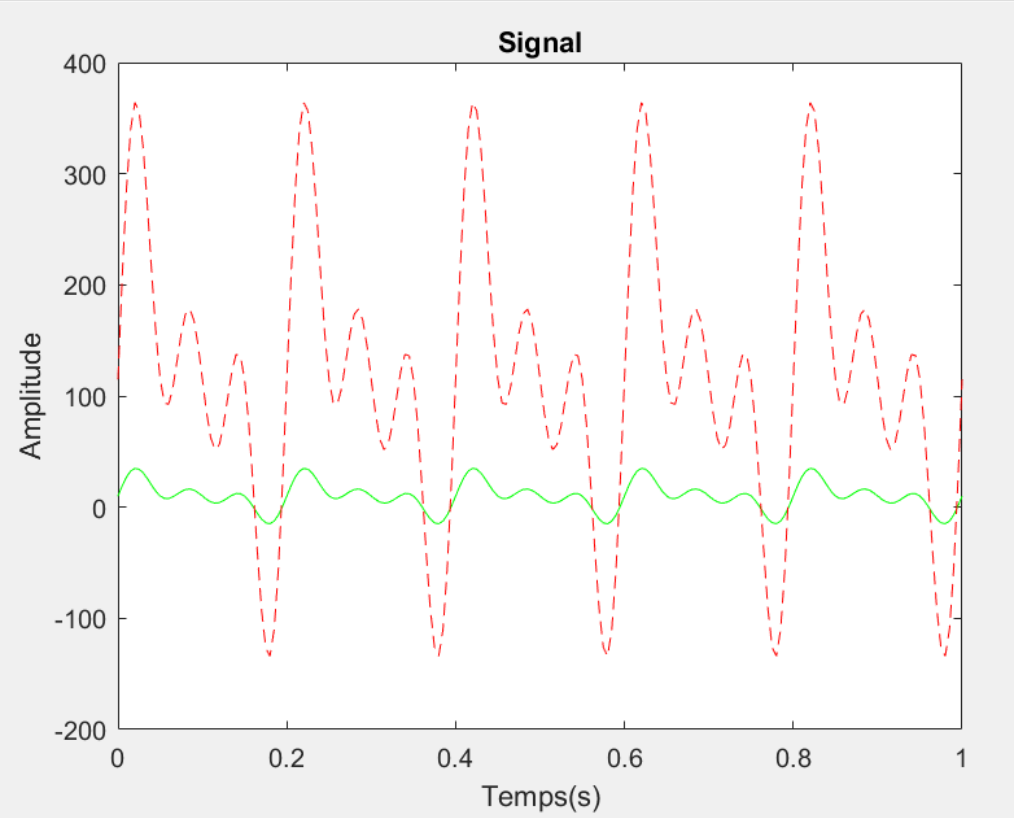

Figure 12 : Fonction du signal y avec l'ajout du gain et de l'offset

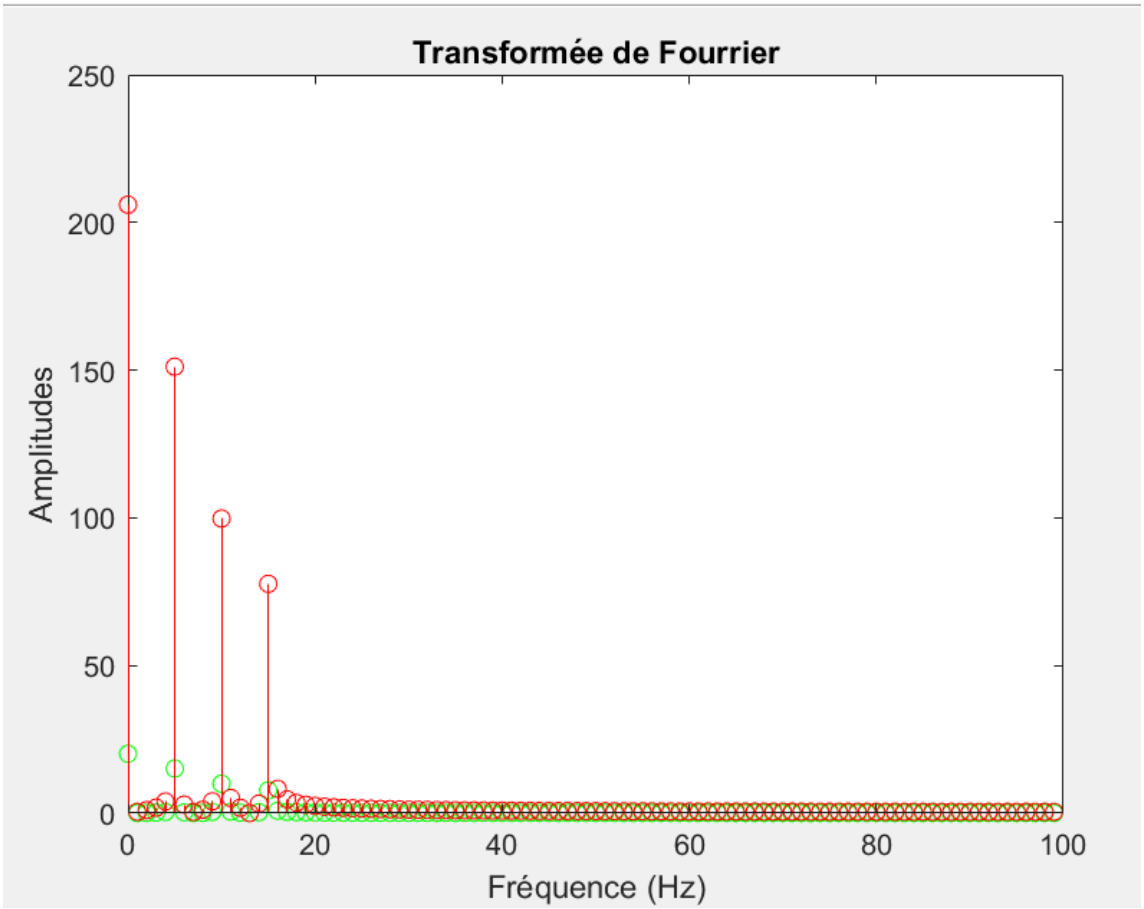

Figure 13 : Graphique de la transformée de Fourrier avec le gain et l'offset

- **Fonction 3 : La non-linéarité**

Dans un troisième temps, nous avons créé la fonction non-linéarité correspondante à notre erreur, en rajoutant des nouveaux paramètre lambda, mu et mu1.

```
y=M+A1*sin(2*pi*f1*t)+ A2*sin(2*pi*f2*t) + A3*sin(2*pi*f3*t);

%Signal de base de fourier
Y=abs(fft(y))/fe;

%signal de base avec erreur linéaire sous fourier
se= G * y + O;
vf=[0:1:fe];
Ynew=[Y(1,1), 2*Y(1,2:floor(fe/2))];%notre moyenne est Y(1,1) cette ligne pour ajuster tout les amplitudes à 10
SE=abs(fft(se))/fe;

%Erreur polynôme d'ordre 2
d=a*t.^2+b*t+c;%Polynôme appliquée à t
j=a*y.^2+b*y+c;%Polynôme appliquée au signal y de base

%Erreur exponentielle
h=lmbd*exp(t/mu1);%Exponentielle appliquée à t
e=lmbd*exp(y/mu1);%Exponentielle appliquée au signal Y de base

%Erreur logarithmique
q=lambda*log(abs(t/mu));%Logarithme appliquée à x
r=lambda*log(abs(y/mu));%Logarithme appliquée au signal Y de base
```

Figure 14 : Non-linéarité

Mise en place de l'IHM

Une fois que nous avons modélisé notre signal harmonique et un spectre temporel, nous avons conçu notre Interface Homme-Machine. Pour ce faire, nous avons placé un slider pour chacune de nos valeurs A1, A2, A3, f1, f2, f3, fe et M. Cela permet à l'utilisateur de choisir les valeurs qu'il souhaite appliquer au système. La valeur de la fréquence d'échantillonnage est importante dans le deuxième graphe car la transformée de Fourrier doit être réalisée selon les critères de Shanon pour avoir un spectre exploitable.

Nous avons un graphique pour notre signal, une pour la transformée de Fourrier et enfin le graphique pour la non-linéarité.

Ainsi, nous présentons notre IHM avec les valeurs des amplitudes initialisées, avec G = 0 et O=0.

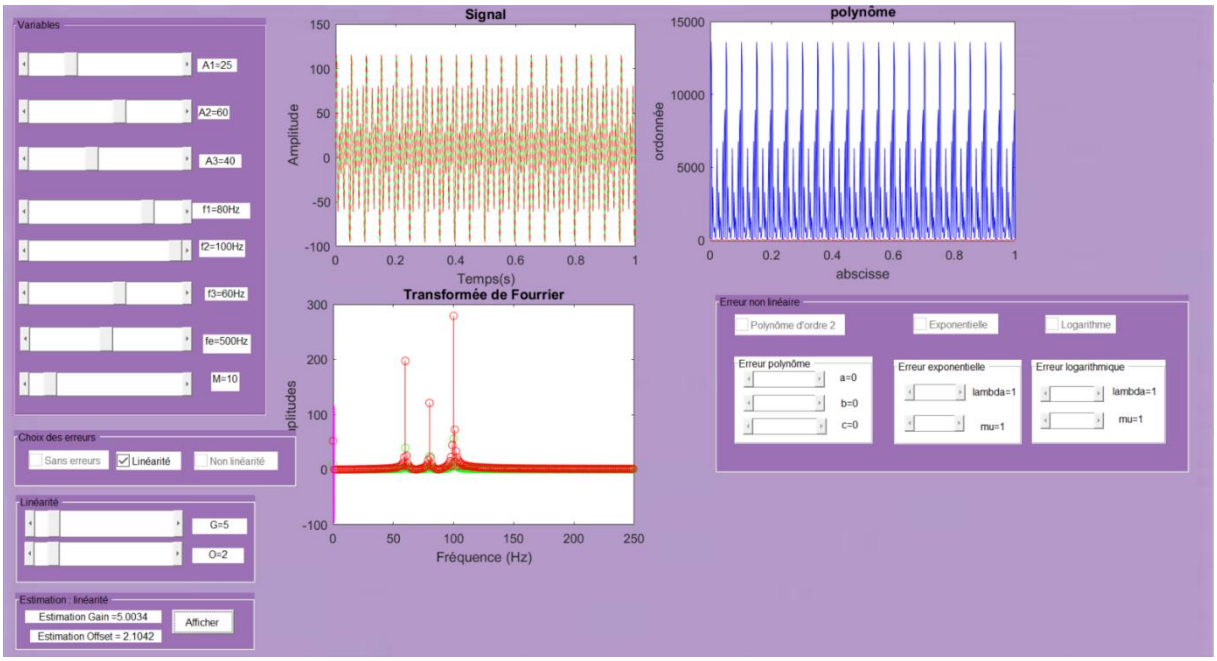

Figure 15 : IHM avec erreur de linéarité

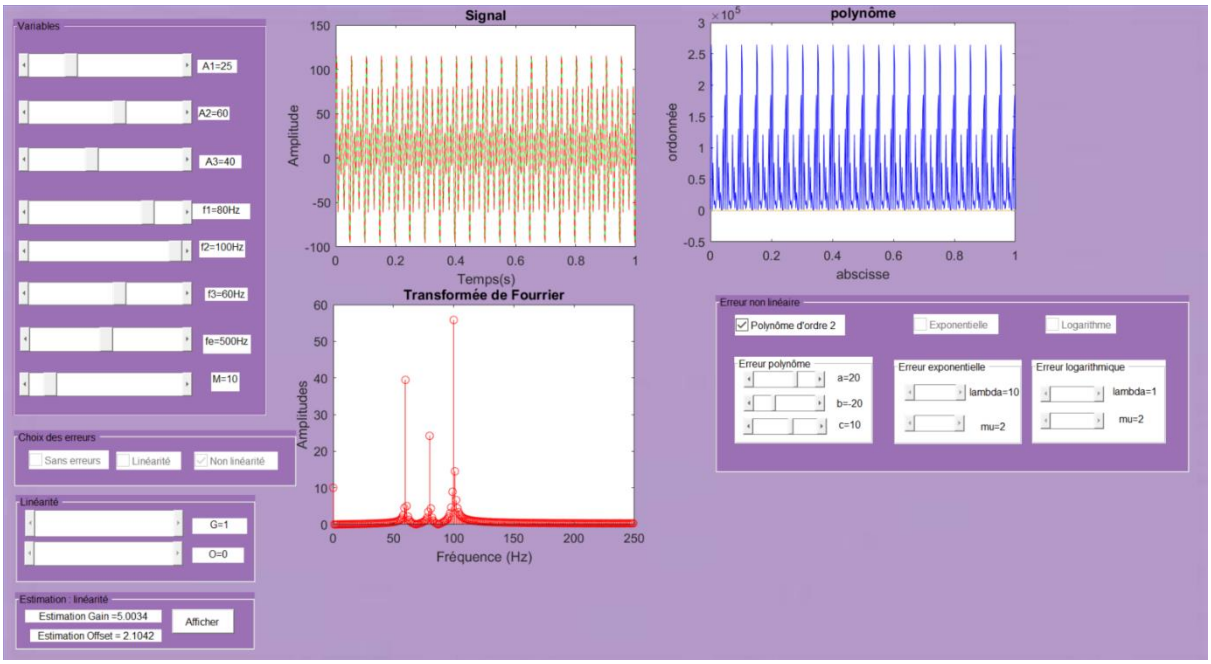

Figure 16 : IHM avec erreur de non-linéarité : cas du polynôme d'ordre 2

Application des erreurs

A l'aide d'une check-box, l'utilisateur va pouvoir choisir un type d'erreur : linéarité et non-linéarité, tout en lui laissant le choix de simuler un signal sans erreur. Nous avons également ajouté deux sliders permettant de faire varier l'offset et le gain.

a- Application de l'erreur de gain et de l'offset

Dans le cas de l'erreur de gain, nous avons appliqué à notre signal de base une erreur linéaire sous la forme de f(x) = ax + b.

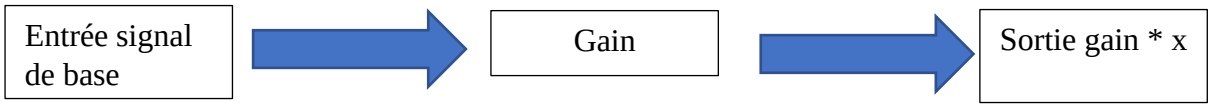

Figure 17 : Schéma résumant l'application du gain à un signal

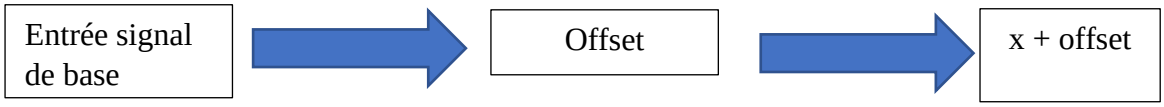

Figure 18 : Schéma résumant l'application de l'offset à un signal

Figure 19 : Récapitulatif de l'application simultanée du gain et de l'offset

b- Application de l'erreur de non-linéarité

Pour cette partie, nous avons appliqué à notre signal une fonction non linéaire. Nous avons choisi trois fonctions linéaires :

- Exponentielle : appliquée au vecteur de temps : **h = lmbd*exp(t/mu1)** et au signal de base y : **e = lmbd*exp(y/mu1)**

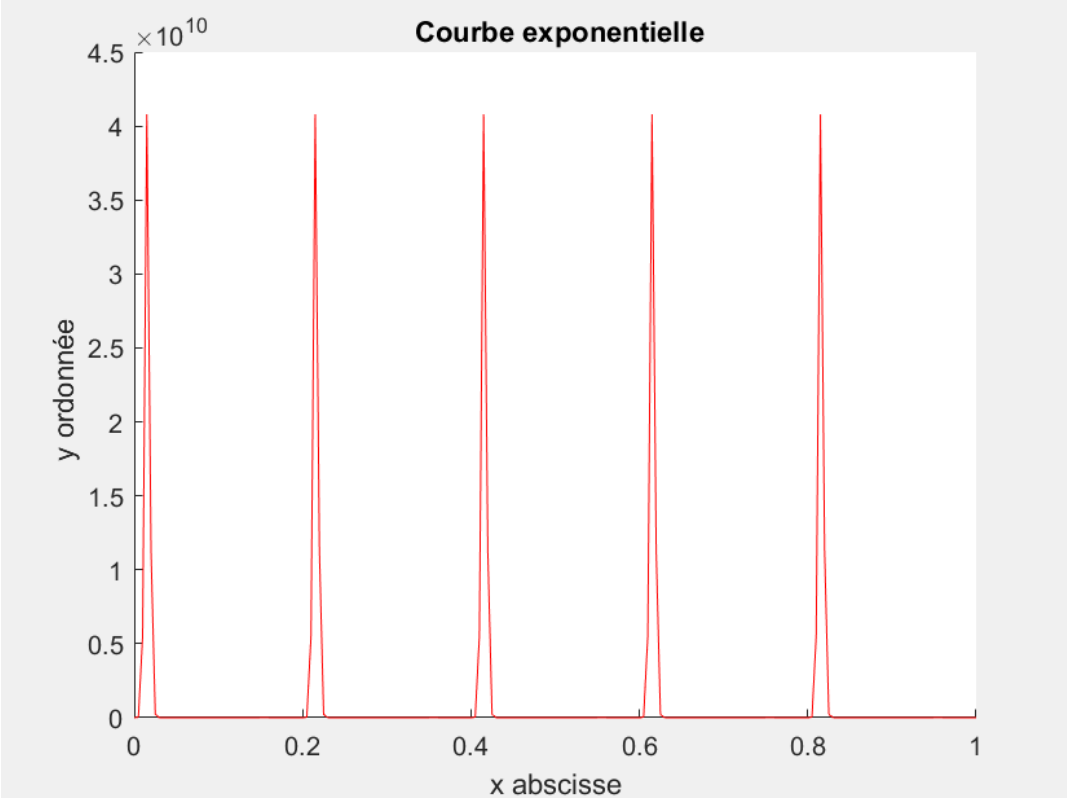

Figure 20 : Signal non linéaire dans le cas de la fonction exponentielle

- Polynôme d'ordre 2 : appliqué au vecteur de temps **: d = a*t.^2+b*t+c** et au signal de base y : **j = a*y.^2+b*y+c**

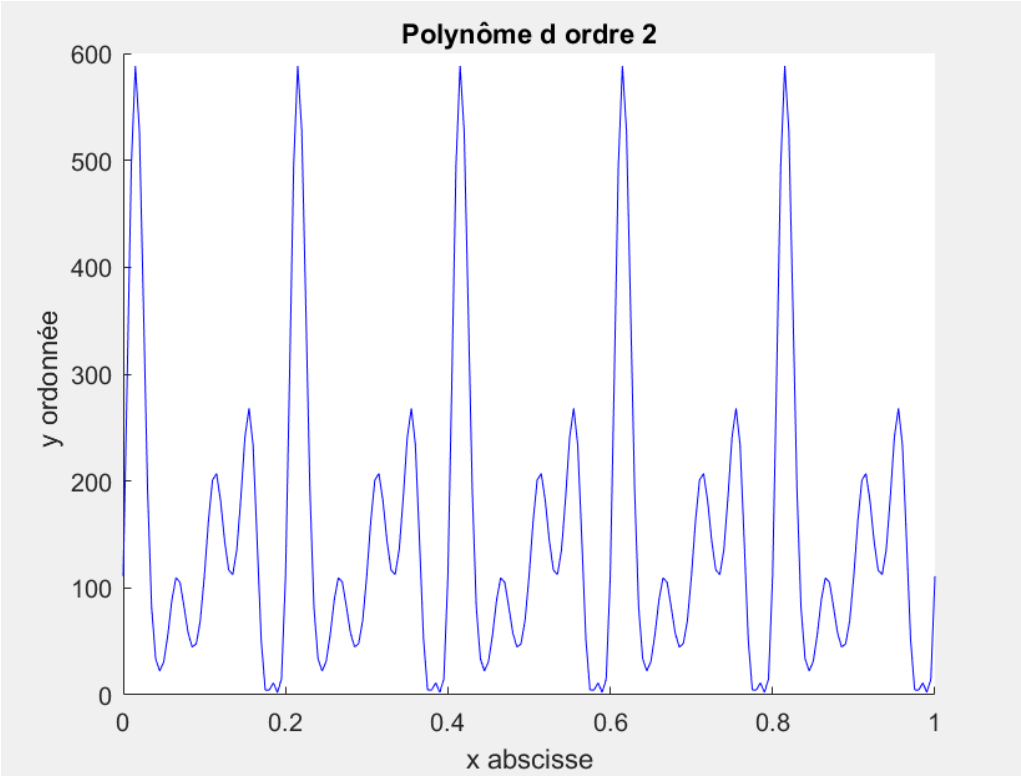

Figure 21 : Signal non linéaire dans le cas de la fonction polynôme d'ordre 2

- Logarithmique : appliquée à x : **q = lambda*log(abs(t/mu))** et au signal de base : **r = lambda*log(abs(y/mu))**

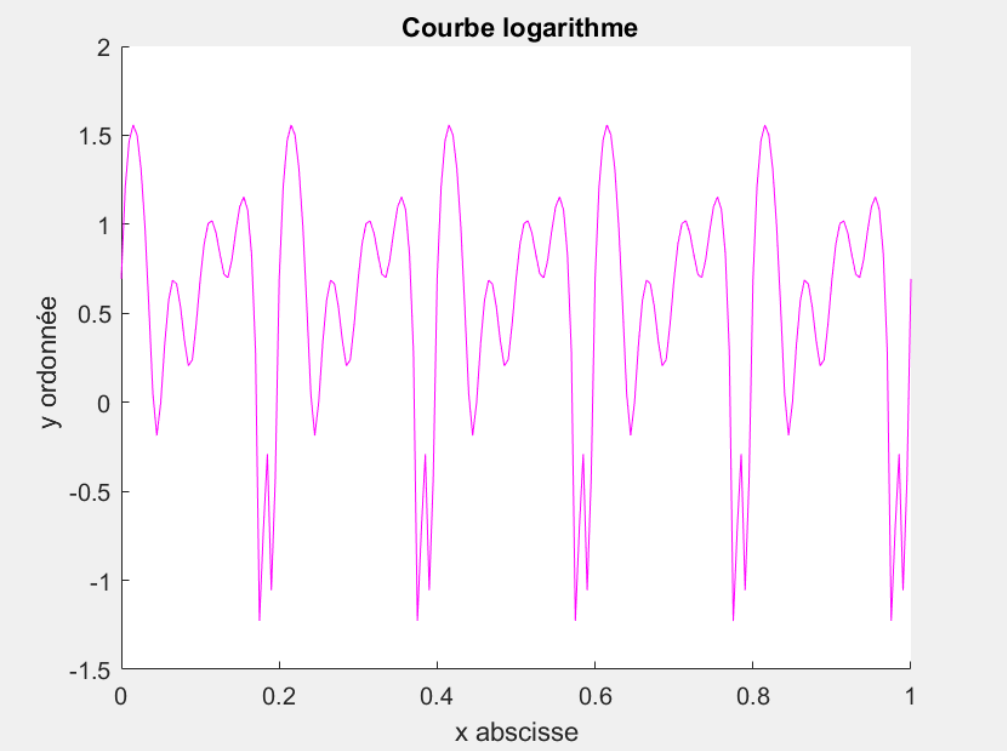

Figure 22 : Signal non linéaire dans le cas de la fonction logarithme

Nous avons donc une fonction j correspondant à notre fonction y que nous avons élevé au second degré. Nous avons choisi trois constantes a, b et c. Nous avons ensuite fait la même chose avec l'erreur de type exponentielle en choisissant λ et μ.

Difficultés rencontrées

Durant la réalisation de ce projet, nous avons rencontré plusieurs difficultés :

- L'extraction des données expérimentales concernant les variables λ, μ, gain et offset ainsi que l'estimation de leurs erreurs.
- L'organisation de notre code : nous n'avons pas tout de suite transformé notre code sous forme de fonction, ce qui nous a ralentie dans la progression de notre projet.

Conclusion

L'objectif de projet est de programmer une interface Homme-Machine, qui comportait un signal ainsi qu'une transformée de Fourrier auxquels nous avons pu ajouter différents types d'erreurs.

Ce projet nous a permis d'approfondir davantage nos connaissances sur Matlab. Nous avons pu grâce à ce second projet sur cette plateforme, d'utiliser de nouvelles fonctions et de les assimiler. Par ailleurs, cela nous a permis de mettre en pratique les notions vues en cours pour mieux les comprendre : repérer les erreurs qu'un signal généré par un capteur peut subir.

Nous avons acquis des compétences professionnelles telles que la cohésion de groupe ainsi que l'utilisation de Matlab, qui nous sera utile dans notre future carrière.

Nous avons pu surmonter les difficultés rencontrées grâce au travail de l'équipe.

A l'avenir, on pourrait améliorer l'ergonomie de notre IHM, réaliser l'extraction des données pour les variables λ, μ, gain et offset dans un tableau.

Evaluation de l'enseignement

Ce cours nous a permis d'améliorer nos compétences de programmation sous Matlab. Par moment, il nous est arrivé de ne pas savoir appliquer immédiatement certaines notions ou certains attendus. Cependant, grâce à la pédagogie et l'écoute de M. FOURNIER, nous avons réussi à rester concentrées et investies dans ce projet. C'était encourageant pour la suite du projet.

Glossaire

Abs(x) : Elle renvoie la valeur absolue d'une scalaire ou de chaque élément du tableau ou d'un vecteur X.

Clc : Efface la fenêtre de commande

Clear all : Elle permet de supprimer toutes les variables du workspace actuel.

Close all : Cette fonction MATLAB permet de fermer tous les profils qui sont ouverts

Global : Déclarer les variables comme globales

Grid on : afficher une grille

Hold on : Conservation des tracés dans les axes actuels afin que les nouveaux tracés ajoutés aux axes ne suppriment pas les tracés existants.

Plot : Trace des courbes en insérant des vecteurs de même longueur dans la fonction.

Stem(x) : Cette fonction MATLAB permet de tracer la séquence de donnée, sous la forme de tiges.

Title : afficher le titre

Xlabel : Afficher les abscisses

Ylabel : afficher les ordonnées

Annexe

Code

```
function [se,t,y]=fonction_signal(M,A1,A2,A3,f1,f2,f3,fe,G,O)

Te=1/fe
t=[0:Te:1];

y=M+A1*sin(2*pi*f1*t)+ A2*sin(2*pi*f2*t) + A3*sin(2*pi*f3*t);%fonction
% if AFF
%     if G==0 & O==0
%     plot(t,y,'g')
%     title('Signal')
%     xlabel('Temps(s)')
%     ylabel('Amplitude')
%     end
% end
%selection de la courbe
Y=abs(fft(y))/fe

% GAIN ET OFFSET
%on ajoute le gain au niveau du graph 1 (superposition des deux courbes)

se= G * y + O
vf=[0:1:fe]
Ynew=[Y(1,1), 2*Y(1,2:floor(fe/2))]
% if AFF
% hold on
% plot(t,se,'--r')
% hold off
% end
 %notre moyenne est Y(1,1) cette ligne pour ajuster tout les amplitudes à 10

end
```

```
function [Ynew,Ybis,t,y,vf]=fonction_TF(M,A1,A2,A3,f1,f2,f3,fe,G,O)

Te=1/fe
t=[0:Te:1];
y=M+A1*sin(2*pi*f1*t)+ A2*sin(2*pi*f2*t) + A3*sin(2*pi*f3*t);

%Transformé de fourrier
Y=abs(fft(y))/fe;

%GAIN ET OFFSET
%on ajoute le gain au niveau du graph 1 (superposition des deux courbes)
se= G * y + O;
% x=t*fe
vf=[0:1:fe];
Ynew=[Y(1,1), 2*Y(1,2:floor(fe/2))];%notre moyenne est Y(1,1) cette ligne pour ajuster tout les amplitudes à 10

SE=abs(fft(se))/fe;

% if AFF
% stem(vf(1,1:floor(fe/2)),Ynew, 'g')
% title('Transformée de Fourrier')
% xlabel('Fréquence (Hz)')
% ylabel('Amplitudes')
% end

%Offset et gain pour le TF

vf=[0:1:fe];

Ybis=[SE(1,1), 2*SE(1,2:floor(fe/2))]

% if AFF
% hold on
% stem(vf(1,1:floor(fe/2)),Ybis(1,1:floor(fe/2)),'r')
% hold off
% end
end
```

```
function [t,y,vf,Ynew,SE,se,d,j,h,e,q,r]= fonction_non_lineaire(M,A1,A2,A3,f1,f2,f3,fe,G,O,a,b,c,lmbd,lambda,mu,mu1)
%on a un signal auquel on va appliquer différentes erreurs
%Entrées:

%A1: Amplitude 1
%A2: Amplitude 2
%A3: Amplitude 3

%f1: fréquence 1
%f2: fréquence 2
%f3: fréquence 3
%fe: fréquence d'échantillonnage

%G: Gain (erreur linéaire)
%O: Offset (erreur linéaire)

%a: paramètre a du polynôme d'ordre 2 (erreur non linéaire)
%b: paramètre b du polynôme d'ordre 2 (erreur non linéaire)
%c: paramètre c du polynôme d'ordre 2 (erreur non linéaire)

%lmbd: paramètre lambda de l'exponentielle (erreur non linéaire)
%mu1:paramètre mu de l'exponentielle (erreur non linéaire)
%mu: paramètre mu du logarithme (erreur non linéaire)
%lambda: paramètre lambda du logarithme (erreur non linéaire)

% Sortie:
% t: vecteur allant de 0 à 1 avec un pas de Te
% vf: vecteur fréquence allant de 0 à fe avec un pas de 1
% y: Signal sinusoïdal de sortie
% Ynew: Fourier du signal y sans erreur
% Ybis:Fourier du signal y avec erreur
% se: Signal sinusoïdal auquel une erreur linéaire a été appliquée (gain et offset)
% SE: Fourier du Signal sinusoïdal auquel une erreur linéaire a été appliquée (gain et offset)

% d: Polynôme appliquée à t
% j: Polynôme appliquée au signal de base y
% h: Exponentielle appliquée à t
% e: Exponentielle appliquée au signal de base y
% q: Logarithme appliquée à t
% r: Logarithme appliquée au signal de base y

Te=1/fe
t=[0:Te:1];
% vecteur fréquence
vf=[0:1:fe];

% Signal de base
y=M+A1*sin(2*pi*f1*t)+ A2*sin(2*pi*f2*t) + A3*sin(2*pi*f3*t);

%Signal de base de fourier
Y=abs(fft(y))/fe;

%signal de base avec erreur linéaire sous fourier
se= G * y + O;
vf=[0:1:fe];
Ynew=[Y(1,1), 2*Y(1,2:floor(fe/2))];%notre moyenne est Y(1,1) cette ligne pour ajuster tout les amplitudes à 10
SE=abs(fft(se))/fe;

%Erreur polynôme d'ordre 2
d=a*t.^2+b*t+c;%Polynôme appliquée à t
j=a*y.^2+b*y+c;%Polynôme appliquée au signal y de base

%Erreur exponentielle
h=lmbd*exp(t/mu1);%Exponentielle appliquée à t
e=lmbd*exp(y/mu1);%Exponentielle appliquée au signal Y de base

%Erreur logarithmique
q=lambda*log(abs(t/mu));%Logarithme appliquée à x
r=lambda*log(abs(y/mu));%Logarithme appliquée au signal Y de base

% % if AFF
%     hold on;
%     plot(t,r,'b')
% %     title(' Courbe logarithme')
```

```
% Signal de base
y=M+A1*sin(2*pi*f1*t)+ A2*sin(2*pi*f2*t) + A3*sin(2*pi*f3*t);

%Signal de base de fourier
Y=abs(fft(y))/fe;

%signal de base avec erreur linéaire sous fourier
se= G * y + O;
vf=[0:1:fe];
Ynew=[Y(1,1), 2*Y(1,2:floor(fe/2))];%notre moyenne est Y(1,1) cette ligne pour ajuster tout les amplitudes à 10
SE=abs(fft(se))/fe;

%Erreur polynôme d'ordre 2
d=a*t.^2+b*t+c;%Polynôme appliquée à t
j=a*y.^2+b*y+c;%Polynôme appliquée au signal y de base

%Erreur exponentielle
h=lmbd*exp(t/mu1);%Exponentielle appliquée à t
e=lmbd*exp(y/mu1);%Exponentielle appliquée au signal Y de base

%Erreur logarithmique
q=lambda*log(abs(t/mu));%Logarithme appliquée à x
r=lambda*log(abs(y/mu));%Logarithme appliquée au signal Y de base

% % if AFF
%     hold on;
%     plot(t,r,'b')
% %     title(' Courbe logarithme')
% %     xlabel('x abscisse')
% %     ylabel('y ordonnée ')
%     hold on;
%     plot(t,y,'m')
%     plot(t,d)
%     hold off
end
```

```
% --- Executes just before IHM is made visible.
function IHM_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to IHM (see VARARGIN)

% Choose default command line output for IHM
handles.output = hObject;

%afficher un fond
ah = axes('unit', 'normalized', 'position', [0 0 1 1]);
bg = imread('photo1.jpg'); imagesc(bg);
set(ah,'handlevisibility','off','visible','off')


% Update handles structure
guidata(hObject, handles);

global A1
global A2
global A3
global f1
global f2
global f3
global fe
global G
global O
global M
global a
global b
global c
global lmbd
global lambda
global mu
global mu1
```

```
global t
global vf
global y
global SE
global se
global d
global j
global h
global e
global q
global r
global Ybis
global Ynew


%On fait appel au fichier init pour avoir des valeurs de départ
run('init1.m');

%on positionne les sliders selon la valeur récupérée précédemment
set(handles.A1,'Value', A1)
set(handles.A2,'Value', A2)
set(handles.A3,'Value', A3)
set(handles.fe,'Value', fe)
set(handles.f1,'Value', f1)
set(handles.f2,'Value', f2)
set(handles.f3,'Value', f3)
set(handles.gain,'Value', G)
set(handles.offset,'Value', O)
set(handles.M,'Value', M)
```

```
%notation de la valeur initiale des variables dans leur encadré
set(handles.text_A1,'string',['A1=',num2str(A1)])
set(handles.text_A2,'string',['A2=',num2str(A2)])
set(handles.text_A3,'string',['A3=',num2str(A3)])
set(handles.text_fe,'string',['fe=',num2str(fe),'Hz'])
set(handles.text_f1,'string',['f1=',num2str(f1),'Hz'])
set(handles.text_f2,'string',['f2=',num2str(f2), 'Hz'])
set(handles.text_f3,'string',['f3=',num2str(f3), 'Hz'])
set(handles.text_g,'string',['G=',num2str(G)])
set(handles.text_o,'string',['O=',num2str(O)])
set(handles.text_m,'string',['M=',num2str(M)])
set(handles.text_a,'string',['a=',num2str(a)])
set(handles.text_b,'string',['b=',num2str(b)])
set(handles.text_c,'string',['c=',num2str(c)])
set(handles.text_lambda1,'string',['lambda=',num2str(lmbd)])
set(handles.text_mu_log,'string',['mu=',num2str(mu)])
set(handles.text_mu,'string',['mu=',num2str(mu1)])
set(handles.text_lambda,'string',['lambda=',num2str(lambda)])

%On définit les slider qui sont accessibles à l'utilisateur à l'ouverture
%de l'IHM
set(handles.gain,'Enable','Off');
set(handles.offset,'Enable','Off');
set(handles.a,'Enable','Off');
set(handles.b,'Enable','Off');
set(handles.c,'Enable','Off');
set(handles.checkbox4,'Enable','Off');
set(handles.checkbox5,'Enable','Off');
set(handles.checkbox6,'Enable','Off');
set(handles.lambda_exp,'Enable','Off');
set(handles.mu_exp,'Enable','Off');
set(handles.lambda_log,'Enable','Off');
set(handles.mu_log,'Enable','Off');
```

```
[se,t,y]=fonction_signal(M,A1,A2,A3,f1,f2,f3,fe,G,O)%signal

[Ynew,Ybis,t,y,vf]=fonction_TF(M,A1,A2,A3,f1,f2,f3,fe,G,O)%fonction transformée de fourier

[t,y,vf,Ynew,SE,se,d,j,h,e,q,r]= fonction_non_lineaire(M,A1,A2,A3,f1,f2,f3,fe,G,O,a,b,c,lmbd,lambda,mu,mu1)%fonction non linéaire

%affichage du premier graphique
axes(handles.axes1)
%on ajoute une condition pour afficher la courbe sans erreur et avec erreur
%selon les checkboxs cochés
    if G==0 & O==0
        plot(t,y,'g')
        title('Signal')
        xlabel('Temps(s)')
        ylabel('Amplitude')
    else
        plot(t,y,'g')
        title('Signal')
        xlabel('Temps(s)')
        ylabel('Amplitude')
        hold on
        plot(t,y,'--r')
        title('Signal')
        xlabel('Temps(s)')
        ylabel('Amplitude')
        hold off
    end
```

```
%affichage du second graphique
axes(handles.axes2)
    if G==0 & O==0
        stem(vf(1,1:floor(fe/2)),Ynew, 'g')
        title('Transformée de Fourrier')
        xlabel('Fréquence (Hz)')
        ylabel('Amplitudes')

    else
        stem(vf(1,1:floor(fe/2)),Ynew, 'g')
        hold on
        stem(vf(1,1:floor(fe/2)),Ybis(1,1:floor(fe/2)),'r')
        title('Transformée de Fourrier')
        xlabel('Fréquence (Hz)')
        ylabel('Amplitudes')
        hold off
    end
%affichage du troisième graphique
axes(handles.axes3)
plot(t,j,'b')
hold on;
plot(t,y,'m')
plot(t,d)
hold off
```

```
O=get(hObject,'Value');
% set(handles.offset,'Value', O)
set(handles.text_o,'string',['O=',num2str(O)])

[se,t,y]=fonction_signal(M,A1,A2,A3,f1,f2,f3,fe,G,O)%signal

[Ynew,Ybis,t,y,vf]=fonction_TF(M,A1,A2,A3,f1,f2,f3,fe,G,O)

% [t,y,vf,Ynew,SE,se,d,j,h,e,q,r]= fonction_non_lineaire(M,A1,A2,A3,f1,f2,f3,fe,G,O,a,b,c,lmbd,lambda,mu,mu1)
```

```
A1=get(hObject,'Value');
% set(handles.A1,'Value', A1)
set(handles.text_A1,'string',['A1=',num2str(A1)])

[se,t,y]=fonction_signal(M,A1,A2,A3,f1,f2,f3,fe,G,O)%signal

[Ynew,Ybis,t,y,vf]=fonction_TF(M,A1,A2,A3,f1,f2,f3,fe,G,O)

[t,y,vf,Ynew,SE,se,d,j,h,e,q,r]= fonction_non_lineaire(M,A1,A2,A3,f1,f2,f3,fe,G,O,a,b,c,lmbd,lambda,mu,mu1)
```

```
A2=get(hObject,'Value');
set(handles.A2,'Value', A2)
set(handles.text_A2,'string',['A2=',num2str(A2)])
```

```
A3=get(hObject,'Value');
% set(handles.A3,'Value', A3)
set(handles.text_A3,'string',['A3=',num2str(A3)])
```

```
f1=get(hObject,'Value');
set(handles.f1,'Value', f1)
set(handles.text_f1,'string',['f1=',num2str(f1),'Hz'])
```

```
f2=get(hObject,'Value');
% set(handles.f2,'Value', f2)
set(handles.text_f2,'string',['f2=',num2str(f2),'Hz'])
```

```
f3=get(hObject,'Value');
% set(handles.f3,'Value', f3)
set(handles.text_f3,'string',['f3=',num2str(f3),'Hz'])
```

```
G=get(hObject,'Value');
% set(handles.gain,'Value', G)
set(handles.text_g,'string',['G=',num2str(G)])
```

```
fe=floor(get(hObject,'Value'));
% set(handles.fe,'Value', fe)
set(handles.text_fe,'string',['fe=',num2str(fe),'Hz'])
```

```
% set(handles.M,'Value', M)
M=floor(get(hObject,'Value'));
set(handles.text_m,'string',['M=',num2str(M)])
```

```
Check=get(handles.checkbox1,'Value')
if Check==1
    %si l'utilisateur coche la case sans erreur alors on dé
    %sliders accessibles ou non
    set(handles.gain,'Enable','Off');
    set(handles.offset,'Enable','Off');
    set(handles.a,'Enable','Off');
    set(handles.b,'Enable','Off');
    set(handles.c,'Enable','Off');
    set(handles.checkbox3,'Enable','Off');
    set(handles.checkbox2,'Enable','Off');
    set(handles.checkbox4,'Enable','Off');
    set(handles.checkbox5,'Enable','Off');
    set(handles.checkbox6,'Enable','Off');
    set(handles.lambda_exp,'Enable','Off');
    set(handles.mu_exp,'Enable','Off');
    set(handles.lambda_log,'Enable','Off');
    set(handles.mu_log,'Enable','Off');

    G=0;
    set(handles.text_g,'string',['G=',num2str(G)])

    O=0;
    set(handles.text_o,'string',['O=',num2str(O)])
    a=0;
    set(handles.text_a,'string',['a=',num2str(a)])

    b=0;
    set(handles.text_b,'string',['b=',num2str(b)])

    c=0;
    set(handles.text_c,'string',['c=',num2str(c)])
```

```
    axes(handles.axes1)
    if G==0 & O==0
        plot(t,y,'g')
        title('Signal')
        xlabel('Temps(s)')
        ylabel('Amplitude')
    else
        plot(t,y,'g')
        title('Signal')
        xlabel('Temps(s)')
        ylabel('Amplitude')
        hold on
        plot(t,y,'--r')
        title('Signal')
        xlabel('Temps(s)')
        ylabel('Amplitude')
        hold off
    end

axes(handles.axes2)
    if G==0 & O==0
        stem(vf(1,1:floor(fe/2)),Ynew, 'g')
        title('Transformée de Fourrier')
        xlabel('Fréquence (Hz)')
        ylabel('Amplitudes')

    else
        stem(vf(1,1:floor(fe/2)),Ynew, 'g')
        hold on
        stem(vf(1,1:floor(fe/2)),Ybis(1,1:floor(fe/2)),'r')
        title('Transformée de Fourrier')
        xlabel('Fréquence (Hz)')
        ylabel('Amplitudes')
        hold off
    end
```

```
    axes(handles.axes3)
    plot(t,j,'b')
    hold on;
    plot(t,y,'m')
    plot(t,d)
    title('polynôme')
    xlabel('abscisse')
    ylabel ('ordonnée')
    hold off
else
    set(handles.gain,'Enable','Off');
    set(handles.offset,'Enable','Off');
    set(handles.a,'Enable','Off');
    set(handles.b,'Enable','Off');
    set(handles.c,'Enable','Off');
    set(handles.checkbox3,'Enable','On');
    set(handles.checkbox2,'Enable','On');
    set(handles.checkbox4,'Enable','Off');
    set(handles.checkbox5,'Enable','Off');
    set(handles.checkbox6,'Enable','Off');
    set(handles.lambda_exp,'Enable','Off');
    set(handles.mu_exp,'Enable','Off');
    set(handles.lambda_log,'Enable','Off');
    set(handles.mu_log,'Enable','Off');

end

Check=get(handles.checkbox2,'Value')
if Check==1
    set(handles.gain,'Enable','On');
    set(handles.offset,'Enable','On');
    set(handles.a,'Enable','Off');
    set(handles.b,'Enable','Off');
    set(handles.c,'Enable','Off');
    set(handles.checkbox3,'Enable','Off');
    set(handles.checkbox1,'Enable','Off');
%     set(handles.checkbox4,'Enable','Off');
%     set(handles.checkbox5,'Enable','Off');
%     set(handles.checkbox6,'Enable','Off');
    set(handles.lambda_exp,'Enable','Off');
    set(handles.mu_exp,'Enable','Off');
    set(handles.lambda_log,'Enable','Off');
    set(handles.mu_log,'Enable','Off');

    G=1;
    set(handles.text_g,'string',['G=',num2str(G)])

    O=0;
    set(handles.text_o,'string',['O=',num2str(O)])
    a=0;
    set(handles.text_a,'string',['a=',num2str(a)])

    b=0;
    set(handles.text_b,'string',['b=',num2str(b)])

    c=0;
    set(handles.text_c,'string',['c=',num2str(c)])
```

```
axes(handles.axes1)
if G==0 & O==0
    plot(t,y,'g')
    title('Signal')
    xlabel('Temps(s)')
    ylabel('Amplitude')
else
    plot(t,y,'g')
    title('Signal')
    xlabel('Temps(s)')
    ylabel('Amplitude')
    hold on
    plot(t,y,'--r')
    title('Signal')
    xlabel('Temps(s)')
    ylabel('Amplitude')
    hold off
end

axes(handles.axes2)
    if G==0 & O==0
        stem(vf(1,1:floor(fe/2)),Ynew, 'g')
        title('Transformée de Fourrier')
        xlabel('Fréquence (Hz)')
        ylabel('Amplitudes')

    else
        stem(vf(1,1:floor(fe/2)),Ynew, 'g')
        hold on
        stem(vf(1,1:floor(fe/2)),Ybis(1,1:floor(fe/2)),'r')
        title('Transformée de Fourrier')
        xlabel('Fréquence (Hz)')
        ylabel('Amplitudes')
        hold off
    end
```

```
        axes(handles.axes3)
        plot(t,j,'b')
        hold on;
        plot(t,y,'m')
        plot(t,d)
        title('polynôme')
        xlabel('abscisse')
        ylabel ('ordonnée')
        hold off
    else
        set(handles.gain,'Enable','Off');
        set(handles.offset,'Enable','Off');
        set(handles.a,'Enable','Off');
        set(handles.b,'Enable','Off');
        set(handles.c,'Enable','Off');
        set(handles.checkbox3,'Enable','On');
        set(handles.checkbox1,'Enable','On');
        set(handles.lambda_exp,'Enable','Off');
        set(handles.mu_exp,'Enable','Off');
        set(handles.lambda_log,'Enable','Off');
        set(handles.mu_log,'Enable','Off');

    end

Check=get(handles.checkbox3,'Value')
if Check==1
    set(handles.gain,'Enable','Off');
    set(handles.offset,'Enable','Off');
    set(handles.a,'Enable','Off');
    set(handles.b,'Enable','Off');
    set(handles.c,'Enable','Off');
    set(handles.checkbox2,'Enable','Off');
    set(handles.checkbox1,'Enable','Off');
    set(handles.checkbox4,'Enable','On');
    set(handles.checkbox5,'Enable','On');
    set(handles.checkbox6,'Enable','On');
    set(handles.lambda_exp,'Enable','Off');
    set(handles.mu_exp,'Enable','Off');
    set(handles.lambda_log,'Enable','Off');
    set(handles.mu_log,'Enable','Off');

    G=1;
    set(handles.text_g,'string',['G=',num2str(G)])

    O=0;
    set(handles.text_o,'string',['O=',num2str(O)])
    a=0;
    set(handles.text_a,'string',['a=',num2str(a)])

    b=0;
    set(handles.text_b,'string',['b=',num2str(b)])

    c=0;
    set(handles.text_c,'string',['c=',num2str(c)])
```

```
axes(handles.axes1)
 if G==0 & O==0
     plot(t,y,'g')
     title('Signal')
     xlabel('Temps(s)')
     ylabel('Amplitude')
 else
     plot(t,y,'g')
     title('Signal')
     xlabel('Temps(s)')
     ylabel('Amplitude')
     hold on
     plot(t,y,'--r')
     title('Signal')
     xlabel('Temps(s)')
     ylabel('Amplitude')
     hold off
 end

 axes(handles.axes2)
 if G==0 & O==0
     stem(vf(1,1:floor(fe/2)),Ynew, 'g')
     title('Transformée de Fourrier')
     xlabel('Fréquence (Hz)')
     ylabel('Amplitudes')

 else
     stem(vf(1,1:floor(fe/2)),Ynew, 'g')
     hold on
     stem(vf(1,1:floor(fe/2)),Ybis(1,1:floor(fe/2)),'r')
     title('Transformée de Fourrier')
     xlabel('Fréquence (Hz)')
     ylabel('Amplitudes')
     hold off
 end
```

```
    axes(handles.axes3)
    plot(t,j,'b')
    hold on;
    plot(t,y,'m')
    plot(t,d)
    title('polynôme')
    xlabel('abscisse')
    ylabel ('ordonnée')
    hold off
else
    set(handles.gain,'Enable','Off');
    set(handles.offset,'Enable','Off');
    set(handles.a,'Enable','Off');
    set(handles.b,'Enable','Off');
    set(handles.c,'Enable','Off');
    set(handles.checkbox2,'Enable','On');
    set(handles.checkbox1,'Enable','On');
    set(handles.checkbox4,'Enable','Off');
    set(handles.checkbox5,'Enable','Off');
    set(handles.checkbox6,'Enable','Off');
    set(handles.lambda_exp,'Enable','Off');
    set(handles.mu_exp,'Enable','Off');
    set(handles.lambda_log,'Enable','Off');
    set(handles.mu_log,'Enable','Off');

end
```

```
Check=get(handles.checkbox4,'Value')
if Check==1
    set(handles.gain,'Enable','Off');
    set(handles.offset,'Enable','Off');
    set(handles.a,'Enable','On');
    set(handles.b,'Enable','On');
    set(handles.c,'Enable','On');
    set(handles.checkbox3,'Enable','Off');
    set(handles.checkbox1,'Enable','Off');
    set(handles.checkbox5,'Enable','Off');
    set(handles.checkbox6,'Enable','Off');

    G=1;
    set(handles.text_g,'string',['G=',num2str(G)])

    O=0;
    set(handles.text_o,'string',['O=',num2str(O)])
    a=1;
    set(handles.text_a,'string',['a=',num2str(a)])

    b=2;
    set(handles.text_b,'string',['b=',num2str(b)])

    c=3;
    set(handles.text_c,'string',['c=',num2str(c)])
```

```
axes(handles.axes1)
if G==0 & O==0
    plot(t,y,'g')
    title('Signal')
    xlabel('Temps(s)')
    ylabel('Amplitude')
else
    plot(t,y,'g')
    title('Signal')
    xlabel('Temps(s)')
    ylabel('Amplitude')
    hold on
    plot(t,y,'--r')
    title('Signal')
    xlabel('Temps(s)')
    ylabel('Amplitude')
    hold off
end

axes(handles.axes2)
if G==0 & O==0
    stem(vf(1,1:floor(fe/2)),Ynew, 'g')
    title('Transformée de Fourrier')
    xlabel('Fréquence (Hz)')
    ylabel('Amplitudes')

else
    stem(vf(1,1:floor(fe/2)),Ynew, 'g')
    hold on
    stem(vf(1,1:floor(fe/2)),Ybis(1,1:floor(fe/2)),'r')
    title('Transformée de Fourrier')
    xlabel('Fréquence (Hz)')
    ylabel('Amplitudes')
    hold off
end


    axes(handles.axes3)
    plot(t,j,'b')
    hold on;
    plot(t,y,'m')
    plot(t,d)
    title('polynôme')
    xlabel('abscisse')
    ylabel ('ordonnée')
    hold off
else
    set(handles.gain,'Enable','Off');
    set(handles.offset,'Enable','Off');
    set(handles.a,'Enable','Off');
    set(handles.b,'Enable','Off');
    set(handles.c,'Enable','Off');
    set(handles.checkbox3,'Enable','On');
    set(handles.checkbox1,'Enable','Off');
    set(handles.checkbox5,'Enable','On');
    set(handles.checkbox6,'Enable','On');

end
```

```
Check=get(handles.checkbox5,'Value')
if Check==1
    set(handles.gain,'Enable','Off');
    set(handles.offset,'Enable','Off');
    set(handles.a,'Enable','Off');
    set(handles.b,'Enable','Off');
    set(handles.c,'Enable','Off');
    set(handles.checkbox3,'Enable','Off');
    set(handles.checkbox1,'Enable','Off');
    set(handles.checkbox4,'Enable','Off');
    set(handles.checkbox6,'Enable','Off');
    set(handles.A1,'Enable','Off');
    set(handles.A2,'Enable','Off');
    set(handles.A3,'Enable','Off');
    set(handles.f1,'Enable','Off');
    set(handles.f2,'Enable','Off');
    set(handles.f3,'Enable','Off');
    set(handles.fe,'Enable','Off');
    set(handles.lambda_exp,'Enable','On');
    set(handles.mu_exp,'Enable','On');
    set(handles.lambda_log,'Enable','Off');
    set(handles.mu_log,'Enable','Off');
    G=1;
    set(handles.text_g,'string',['G=',num2str(G)])

    O=0;
    set(handles.text_o,'string',['O=',num2str(O)])
    a=0;
    set(handles.text_a,'string',['a=',num2str(a)])

    b=0;
    set(handles.text_b,'string',['b=',num2str(b)])

    c=0;
    set(handles.text_c,'string',['c=',num2str(c)])
```

```
axes(handles.axes1)
 if G==0 & O==0
     plot(t,y,'g')
     title('Signal')
     xlabel('Temps(s)')
     ylabel('Amplitude')
 else
     plot(t,y,'g')
     title('Signal')
     xlabel('Temps(s)')
     ylabel('Amplitude')
     hold on
     plot(t,y,'--r')
     title('Signal')
     xlabel('Temps(s)')
     ylabel('Amplitude')
     hold off
 end

 axes(handles.axes2)
 if G==0 & O==0
     stem(vf(1,1:floor(fe/2)),Ynew, 'g')
     title('Transformée de Fourrier')
     xlabel('Fréquence (Hz)')
     ylabel('Amplitudes')

 else
     stem(vf(1,1:floor(fe/2)),Ynew, 'g')
     hold on
     stem(vf(1,1:floor(fe/2)),Ybis(1,1:floor(fe/2)),'r')
     title('Transformée de Fourrier')
     xlabel('Fréquence (Hz)')
     ylabel('Amplitudes')
     hold off
 end
```

```
    axes(handles.axes3)
    plot(t,e,'b')
    hold on;
    plot(t,y,'m')
    plot(t,h)
    title('exponentielle')
    xlabel('abscisse')
    ylabel ('ordonnée')
    hold off
else
    set(handles.gain,'Enable','Off');
    set(handles.offset,'Enable','Off');
    set(handles.a,'Enable','Off');
    set(handles.b,'Enable','Off');
    set(handles.c,'Enable','Off');
    set(handles.checkbox3,'Enable','On');
    set(handles.checkbox1,'Enable','Off');
    set(handles.checkbox4,'Enable','On');
    set(handles.checkbox6,'Enable','On');
    set(handles.A1,'Enable','On');
    set(handles.A2,'Enable','On');
    set(handles.A3,'Enable','On');
    set(handles.f1,'Enable','On');
    set(handles.f2,'Enable','On');
    set(handles.f3,'Enable','On');
    set(handles.fe,'Enable','On');
    set(handles.lambda_exp,'Enable','Off');
    set(handles.mu_exp,'Enable','Off');
    set(handles.lambda_log,'Enable','Off');
    set(handles.mu_log,'Enable','Off');

end
```

```
Check=get(handles.checkbox6,'Value')
if Check==1
    set(handles.gain,'Enable','Off');
    set(handles.offset,'Enable','Off');
    set(handles.a,'Enable','Off');
    set(handles.b,'Enable','Off');
    set(handles.c,'Enable','Off');
    set(handles.checkbox3,'Enable','Off');
    set(handles.checkbox1,'Enable','Off');
    set(handles.checkbox4,'Enable','Off');
    set(handles.checkbox5,'Enable','Off');
    set(handles.A1,'Enable','Off');
    set(handles.A2,'Enable','Off');
    set(handles.A3,'Enable','Off');
    set(handles.f1,'Enable','Off');
    set(handles.f2,'Enable','Off');
    set(handles.f3,'Enable','Off');
    set(handles.fe,'Enable','Off');
    set(handles.lambda_exp,'Enable','Off');
    set(handles.mu_exp,'Enable','Off');
    set(handles.lambda_log,'Enable','On');
    set(handles.mu_log,'Enable','On');
    G=1;
    set(handles.text_g,'string',['G=',num2str(G)])

    O=0;
    set(handles.text_o,'string',['O=',num2str(O)])
    a=0;
    set(handles.text_a,'string',['a=',num2str(a)])

    b=0;
    set(handles.text_b,'string',['b=',num2str(b)])

    c=0;
    set(handles.text_c,'string',['c=',num2str(c)])
```

```
axes(handles.axes1)
plot(t,y,'g')
hold on
plot(t,y,'--r')
title('Signal')
xlabel('Temps(s)')
ylabel('Amplitude')
hold off

axes(handles.axes2)
stem(vf(1,1:floor(fe/2)),Ynew, 'g')
hold on
stem(vf(1,1:floor(fe/2)),Ynew(1,1:floor(fe/2)),'r')
title('Transformée de Fourrier')
xlabel('Fréquence (Hz)')
ylabel('Amplitudes')
hold off

axes(handles.axes3)
plot(t,r,'b')
hold on;
plot(t,y,'m')
plot(t,q)
title('logarithme')
xlabel('abscisse')
ylabel ('ordonnée')
hold off
```

```
else
    set(handles.gain,'Enable','Off');
    set(handles.offset,'Enable','Off');
    set(handles.a,'Enable','Off');
    set(handles.b,'Enable','Off');
    set(handles.c,'Enable','Off');
    set(handles.checkbox3,'Enable','On');
    set(handles.checkbox1,'Enable','Off');
    set(handles.checkbox4,'Enable','On');
    set(handles.checkbox5,'Enable','On');
    set(handles.A1,'Enable','On');
    set(handles.A2,'Enable','On');
    set(handles.A3,'Enable','On');
    set(handles.f1,'Enable','On');
    set(handles.f2,'Enable','On');
    set(handles.f3,'Enable','On');
    set(handles.fe,'Enable','On');
    set(handles.lambda_exp,'Enable','Off');
    set(handles.mu_exp,'Enable','Off');
    set(handles.lambda_log,'Enable','Off');
    set(handles.mu_log,'Enable','Off');

end
```

```
[se,t,y]=fonction_signal(M,A1,A2,A3,f1,f2,f3,fe,G,O)%signal

[Ynew,Ybis,t,y,vf]=fonction_TF(M,A1,A2,A3,f1,f2,f3,fe,G,O)

[t,y,vf,Ynew,SE,se,d,j,h,e,q,r]= fonction_non_lineaire(M,A1,A2,A3,f1,f2,f3,fe,G

y=M+A1*sin(2*pi*f1*t)+ A2*sin(2*pi*f2*t) + A3*sin(2*pi*f3*t);%fonction
%extraction

Button=get(handles.pushbuttonli,'Value')
%estimation du gain
Gest=Ybis/Ynew
%estimation de l'erreur sur le gain
erreur_G= 100*abs((Gest-G))
%estimation offset
Off_est=(Ybis(1)-Ynew(1))/20

% v=(se(1)-y(1))/2
if Button==1
    set(handles.text14,'String',['Estimation Gain =',num2str(Gest)])
    if Off_est<0.005*max(y)
        set(handles.text15,'String',['Estimation Offset= negligeable'])
    else
        set(handles.text15,'String',['Estimation Offset = ',num2str(Off_est)])
    end
end
```

```
a=get(hObject,'Value')
set(handles.text_a,'string',['a=',num2str(a)])
[t,y,vf,Ynew,SE,se,d,j,h,e,q,r]= fonction_non_lineaire(M,A1,A2,A3,f1,f2,f3,fe,G,O,a,b,c,lmbd,lambda,mu,mu1)

axes(handles.axes3)
plot(t,j,'b')
hold on;
plot(t,y,'m')
plot(t,d)
title('polynôme')
xlabel('abscisse')
ylabel ('ordonnée')
hold off
```

```
b=get(hObject,'Value')
set(handles.text_b,'string',['b=',num2str(b)])
[t,y,vf,Ynew,SE,se,d,j,h,e,q,r]= fonction_non_lineaire(M,A1,A2,A3,f1,f2,f3,fe,G,O,a,b,c,lmbd,lambda,mu,mu1)
axes(handles.axes3)
plot(t,j,'b')
hold on;
plot(t,y,'m')
plot(t,d)
title('polynôme')
xlabel('abscisse')
ylabel ('ordonnée')
hold off
```

```
c=get(hObject,'Value')
set(handles.text_c,'string',['c=', num2str(c)])
[t,y,vf,Ynew,SE,se,d,j,h,e,q,r]= fonction_non_lineaire(M,A1,A2,A3,f1,f2,f3,fe,G,O,a,b,c,lmbd,lambda,mu,mu1)
axes(handles.axes3)
plot(t,j,'b')
hold on;
plot(t,y,'m')
plot(t,d)
title('polynôme')
xlabel('abscisse')
ylabel ('ordonnée')
hold off
```

```
lambda=get(hObject,'Value')
set(handles.text_lambda,'string',['lambda=',num2str(lambda)])
[t,y,vf,Ynew,SE,se,d,j,h,e,q,r]= fonction_non_lineaire(M,A1,A2,A3,f1,f2,f3,fe,G,O,a,b,c,lmbd,lambda,mu,mu1)
axes(handles.axes3)
plot(t,r,'b')
hold on;
plot(t,y,'m')
plot(t,q)
title('logarithme')
xlabel('abscisse')
ylabel ('ordonnée')
hold off
```

```
mu=get(hObject,'Value')
set(handles.text_mu_log,'string',['mu=',num2str(mu)])
[t,y,vf,Ynew,SE,se,d,j,h,e,q,r]= fonction_non_lineaire(M,A1,A2,A3,f1,f2,f3,fe,G,O,a,b,c,lmbd,lambda,mu,mu1)
axes(handles.axes3)
plot(t,r,'b')
hold on;
plot(t,y,'m')
plot(t,q)
title('logarithme')
xlabel('abscisse')
ylabel ('ordonnée')
hold off
```

```
lmbd=get(hObject,'Value')
set(handles.text_lambda1,'string',['lambda=',num2str(lmbd)])
[t,y,vf,Ynew,SE,se,d,j,h,e,q,r]= fonction_non_lineaire(M,A1,A2,A3,f1,f2,f3,fe,G,O,a,b,c,lmbd,lambda,mu,mu1)
axes(handles.axes3)
plot(t,e,'b')
hold on;
plot(t,y,'m')
plot(t,h)
title('exponentielle')
xlabel('abscisse')
ylabel ('ordonnée')
hold off
```

```
mu1=get(hObject,'Value')
set(handles.text_mu,'string',['mu=',num2str(mu1)])
[t,y,vf,Ynew,SE,se,d,j,h,e,q,r]= fonction_non_lineaire(M,A1,A2,A3,f1,f2,f3,fe,G,O,a,b,c,lmbd,lambda,mu,mu1)
axes(handles.axes3)
plot(t,e,'b')
hold on;
plot(t,y,'m')
plot(t,h)
title('exponentielle')
xlabel('abscisse')
ylabel ('ordonnée')
hold off
```