Université Hassiba Benbouali de Chlef
Faculté de Technologie
Département d'Electronique

TRAVAUX PRATIQUES TRAITEMENT DU SIGNAL

Master 2: Electronique
Option : Automatique et contrôle industrielle

TAHRAOUI SOUAAD Maître de Conférences A, HDR

ANNEE UNIVERSITAIRE 2022/2023

Semestre: 1

Unité d'enseignement: UEM 1.1

Matière: TP Traitement du signal

VHS: 22h30 (TP: 1h30)

Crédits: 2

Coefficient: 1

Objectifs de l'enseignement:

Pour le TP TS, Consolider les connaissances acquises pendant le cours de la matière "Traitement du signal" par des travaux pratiques pour mieux comprendre et assimiler le contenu de cette matière.

Connaissances préalables recommandées

L'étudiant devra posséder les connaissances suivantes :

- **Théorie du signal**
- **Les bases mathématiques**
- **Matlab**

Contenu de la matière :

TP 1 – Représentation de signaux et applications de la transformée de Fourier sous Matlab

TP 2 - Filtrage Analogique

TP3- Transformée de Fourrier Discrète

TP 4- Filtrage Numérique RII

TP5- Filtrage Numérique RIF

Mode d'évaluation : 100% évaluation continue

Avant-propos

Ce polycopie est constitué de textes de travaux pratiques (TP) élabores pour étudier pratiquement les propriétés et théorèmes fondamentaux du traitement du signal analogique et numérique des signaux aussi bien dans le domaine temporel qu'en domaine fréquentiel. L'accent y est mis sur l'utilisation pratique du logiciel Matlab pour la simulation des différents aspects du traitement du signal. Ces TP sont destinés aux étudiants de première année master en automatique et informatique industrielle.

Dans cette perspective, les textes de travaux pratiques présentes dans ce polycopie sont constitués d'un rappel théorique suivi d'une série de manipulations numériques. Ces manipulations permettent à l'étudiant de vérifier pratiquement sur Matlab les différentes notions théoriques présentées dans le cours traitement du signal.

Les travaux pratiques du TS présentes dans ce polycopie se déroulent sur un semestre. A l'issu de ces TP, l'étudiant sera en mesure de : Greneter et visualiser des signaux usuels, analogique et numérique, dans le domaine temporel et fréquentiel ; Exécuter l'interprétation géométrique des principales classes de signaux ; Rappeler le théorème de l'échantillonnage, quantification et choisir la fréquence d'échantillonnage ; Identifier et expliquer l'effet de repliement spectral qui pouvant s'écrire avec l'échantillonnage, Maitriser les séries de Fourier : représentations spectrales ; Représenter et analyser le contenu fréquentiel des signaux à l'aide de la transformée de Fourier FFT.

TP 0 :

Prise en main de MatLab

1. Objectif du TP :

Ce TP doit permettre à tout étudiant de maîtriser l'environnement du logiciel de manière à pouvoir travailler de manière autonome.

1.1. Introduction à Matlab

MATLAB est une abréviation de MATrix LABoratory. Écrit à l'origine en Fortran par C. *Moler,* MATLAB est un environnement puissant, complet et facile à utiliser destiné au calcul scientifique. Il s'agit d'un système interactif intégrant calcul numérique et visualisation. C'est un environnement performant, ouvert et programmable qui permet de remarquables gains de productivité et de créativité. MATLAB dispose de plusieurs centaines de fonctions mathématiques, scientifiques et techniques. Son approche matricielle permet de traiter les données sans aucune limitation de taille et de réaliser des calculs numériques et symboliques de façon fiable et rapide. Le logiciel MATLAB comprend le noyau de base MATLAB et des boîtes à outils (toolboxes) composées de fichiers spéciaux (M-files) qui étendent les fonctionnalités du noyau de base. Ce logiciel est très employé dans des disciplines de grande consommatrice de calcul matriciel comme l'automatique,…..

On peut aussi l'utiliser dans de nombreux autres domaines : traitement du signal, optimisation, …etc. A chaque domaine correspond plusieurs boîtes à outils de MATLAB. Certaines instructions de Matlab sont introduites directement à partir du prompt (>>). Ceci est acceptable si la session des instructions est courte et non répétitive. Cependant, la puissance de Matlab vient de la possibilité d'exécuter de longues séquences de commandes stockées dans des fichiers d'extension .m. Le fichier est juste une suite d'instruction et de fonctions Matlab comme celles utilisées à la suite du prompt de Matlab.

MATLAB permet le travail interactif soit en mode commande, soit en mode programmation ; tout en ayant toujours la possibilité de faire des visualisations graphiques. Considéré comme un des meilleurs langages de programmation, MATLAB possède par rapport au C et au fortran les particularités suivantes :

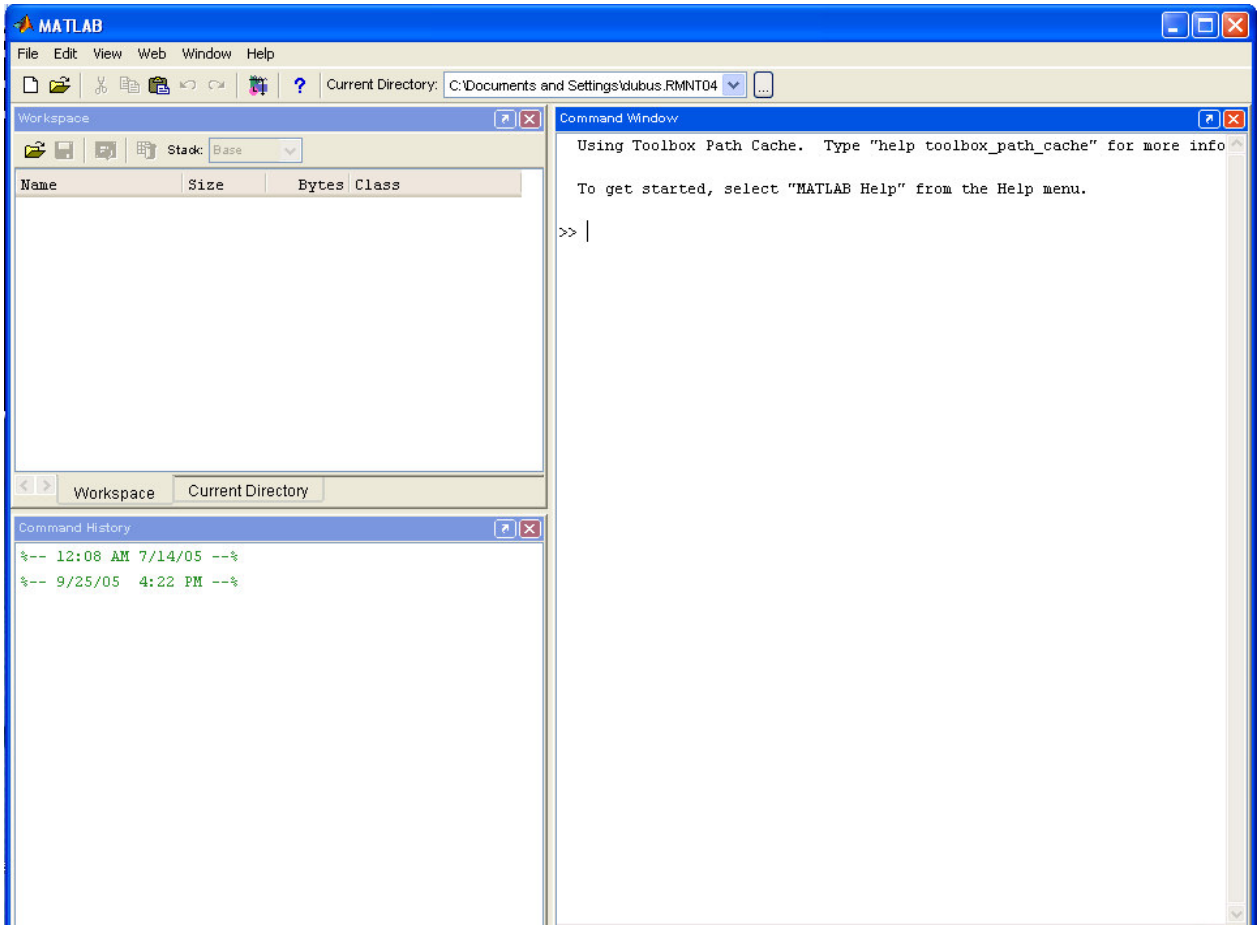
- une programmation facile,
- une continuité parmi les valeurs entières, réelles et complexes,
- une bibliothèque mathématique très compréhensive,
- un outil graphique qui inclut les fonctions d'interface graphique et les utilitaires,

- une possibilité de liaison avec les autres langages classiques de programmation (C ou Fortran).

Aucune déclaration de variables n'est à effectuer. En effet, il n'existe pas de distinction entre les nombres entiers, les nombres réels, les nombres complexes et la simple ou double précision. Cette caractéristique rend le mode de programmation très facile et très rapide.

La bibliothèque des fonctions mathématiques dans MATLAB permet des analyses mathématiques très simples. En effet, l'utilisateur peut exécuter dans le mode commande n'importe quelle fonction mathématique se trouvant dans la bibliothèque sans avoir à recourir à la programmation.

2. Démarrer une session de travail sous MATLAB



Dans la fenêtre Command Window, on tape les instructions une ligne à la fois:

Chaque ligne est exécutée immédiatement après la touche "Return".

Une ligne peut contenir plusieurs instructions séparées par des virgules (,).

2.1. FONCTION "HELP"

Pour obtenir de l'aide sur un sujet, une instruction ou une fonction, on tape help suivi par le sujet, l'instruction ou la fonction désirée.

2.2. ESPACE DE TRAVAIL (Workspace)

Les variables sont définies au fur et à mesure que l'on donne leurs noms et leurs valeurs numériques (ou leurs expressions mathématiques). Les variables ainsi définies sont stockées dans l'espace de travail et peuvent être utilisées dans les calculs subséquents.

2.3. INFORMATION SUR L'ESPACE DE TRAVAIL

Pour obtenir une liste des variables dans l'espace de travail, on utilise les instructions suivantes:

- who : Affichage des variables dans l'espace de travail.
- whos : Affichage détaillé des variables dans l'espace de travail.

2.4. ENREGISTRER LES VARIABLES DE L'ESPACE DE TRAVAIL DANS UN FICHIER

Pour enregistrer les variables de l'espace de travail dans un fichier, on utilise les instructions suivantes:

- save Enregistrer toutes les variables dans un fichier matlab.mat. Dans une session ultérieure, taper load pour ramener l'espace de travail enregistrée.
- save fichier1.mat x y z A X Enregistrer les variables x, y, z, A, X dans le fichier fichier1.mat.Dans une session ultérieure, taper load fichier1 pour ramener les variables x, y, z, A, X dans l'espace de travail.

2.5. Commandes générales :

clear var supprime la variable var de l'espace de travail

clear all supprime toutes les variables

close all ferme tous les graphiques

2.5.1. Lecture et affichage

- **Lecture**

  La commande input permet de demander à l'utilisateur Matlab d'entrer les valeurs de variables à utiliser.

Exp : avec l'instruction :x=input ('valeur de x=')

L'expression « valeur de x = » est affichée à l'écran et matlab attend que l'utilisateur saisisse une donnée au clavier .cette donnée peut être une valeur numérique ou une instruction matlab .

Un retour chariot provoque la fin de la saisie .une valeur numérique est directement affectée à la variable x tandis qu'une instruction Matlab est évaluée et le résultat est affecté à la variable x.

- **L'affichage :**

On peut afficher un message ou une valeur à l'écran avec l'instruction **disp**. Les simples cotes 'indiquent qu'on veut afficher du texte.

Exp : l'instruction :>>disp ('la solution est égale à :') affiche sur l'écran :

La solution est égale à :

3. Opérations mathématiques

3.1. Nombres et opérations arithmétiques

NOMBRES

Les nombres réels peuvent être écrits sous différents formats:
5 1.0237 0.5245E-12 12.78e6 0.001234 -235.087
Les nombres complexes peuvent être écrits sous forme cartésienne ou polaire:
Forme cartésienne: 0.5 + i*2.7 -1.2 + j*0.789 2.5 + 9.7i
Forme polaire: 1.25*exp(j*0.246)

FORMATS D'AFFICHAGE

Pour choisir le format d'affichage pour les nombres, on utilise l'instruction format:

format short 0.1234

format long 0.12345678901234

format short e 1.2341E+002

format long e 0.123456789012345E+002

format hex ABCDEF0123456789

OPÉRATIONS ARITHMÉTIQUES

+ Addition

- Soustraction

* Multiplication

/ Division à droite

\ Division à gauche

^ Puissance

3.2. Vecteurs et matrices

VECTEURS

On peut définir un vecteur x :

- en donnant la liste de ses éléments:

```
>> x=[0.5 1.2 -3.75 5.82 -0.735]
x =
0.5000 1.2000 -3.7500 5.8200 -0.7350
```

- en donnant la suite qui forme le vecteur:

```
>> x=2:0.6:5
x =
2.0000 2.6000 3.2000 3.8000 4.4000 5.0000
```

- en utilisant une fonction qui génère un vecteur:

```
>> x=linspace(1,10,6)
x =
1.0000 2.8000 4.6000 6.4000 8.2000 10.0000
ou:
>> y=logspace(1,3,7)
y =
1.0e+003 *
0.0100 0.0215 0.0464 0.1000 0.2154 0.4642 1.0000
```

Remarque:

Lors qu'on ajoute un «;» à la fin d'une instruction, elle est exécutée mais le résultat n'est pas affiché:

```
>> a=[1 2 3 4 5];

>> b=-2.5;

>> c=b*a;

>>
```

Lors qu'il n'y a pas de «;» à la fin d'une instruction, elle est exécutée et le résultat est affiché:

```
>> a=[1 2 3 4 5]

a =

1 2 3 4 5

>> b=-2.5

b =

-2.5000

>> c=b*a

c =

-2.5000 -5.0000 -7.5000 -10.0000 -12.5000
```

MATRICES

On définit une matrice A en donnant ses éléments:

```
>> A=[0.5 2.7 3.9;4.5 0.85 -1.23;-5.12 2.47 9.03]
A =
0.5000 2.7000 3.9000
4.5000 0.8500 -1.2300
-5.1200 2.4700 9.0300
```

Matrice unitaire:

```
>> B=eye(4)
B =
1 0 0 0
0 1 0 0
0 0 1 0
0 0 0 1
```

EMPLOI DES INDICES

Les éléments d'un vecteur ou d'une matrice peuvent être adressés en utilisant les indices sous la forme suivante:

t(10) élément no. 10 du vecteur t

A(2,9) élément se trouvant à ligne 2, colonne 9 de la matrice A

B(:,7) la colonne 7 de la matrice B

C(3,:) la ligne 3 de la matrice B

OPÉRATIONS MATRICIELLES

Les opérations matricielles exécutées par MATLAB sont illustrées dans le tableau suivant:

B = A' La matrice B est égale à la matrice A transposée
E = inv(A) La matrice E est égale à la matrice A inversée
C = A + B Addition
D = A - B Soustraction
Z = X*Y Multiplication
X = A\B Équivalent à inv(A)*B
X = B/A Équivalent à B*inv(A)

OPÉRATION «ÉLÉMENT PAR ÉLÉMENT»

Les opérations «élément par élément» des vecteurs et des matrices sont effectuées en ajoutant un point (.) avant les opérations * / \ ^ '

Exemple:

```
>> A=[1 2 3 4 5];
>> B=[6 7 8 9 10];
>> C=A.*B
C =
6 14 24 36 50
>> D=A./B
D =
0.1667 0.2857 0.3750 0.4444 0.5000
```

3.3. Variables et fonctions

VARIABLES

On définit une variable en donnant son nom et sa valeur numérique ou son expression mathématique:

```
a =1.25;

x = 0:0.5:10;

y = a*x;

z = y.^2;
```

EXPRESSIONS MATHÉMATIQUES

On écrit les expressions mathématiques de la façon habituelle:

```
z = 5*exp(-0.4*x).*sin(7.5*y);
```

FONCTIONS MATHÉMATIQUES

Les fonctions mathématiques de base sont données dans le tableau suivant:

Fonctions mathématiques élémentaires	
abs	valeur absolu ou module
angle	argument d'un complexe
sqrt	racine carrée
real	partie réelle
imag	partie imaginaire
conj	complexe conjugué
gcd	PGCD
lcm	PPCM
round	arrondi à l'entier le plus proche
fix	troncature
floor	arrondi vers -∞
ceil	arrondi vers +∞
sign	signe de
rem	reste de la division
exp	exponentiel
log	log népérien
log10	log décimal

Les fonctions trigonométriques sont données dans le tableau suivant:

Fonctions trigonométriques	
sin, asin, sinh, asinh	
cos, acos, cosh, acosh	
tan, atan, tanh, atanh	
cot, acot, coth, acoth	
sec, asec, sech, asech	1./cos(z), acos(1./z), 1./cosh(z), acosh(1./z)
csc, acsc, csch, acsch	1./sin(z), asin(1./z), 1./sinh(z), asinh(1./z)

4. CRÉATION DE FONCTIONS

L'usager peut créer des fonctions particulières pour ses applications. Voir «Programmation avec MATLAB». sin cos tan asin acos atan atan2sinh cosh tanh asinh acosh atanh

5. Graphiques sous Matlab

S'il est utile de pouvoir faire des calculs numériques, il est aussi utile d'avoir une représentation graphique des résultats. Pour tracer un graphe en 2D, on fait souvent appel à la fonction Matlab

« **plot** »

5.1. tracer le graphe d'une fonction :

Pour tracer le graphe d'une fonction y en fonction de x, avec x et y des vecteurs de même dimension, on utilise la commande **plot(x,y)** qui ouvre une fenêtre graphique (une figure) et dessine les valeurs de y en ordonnées et celles de x en abscisses.

5.2. Améliorer la lisibilité d'une figure :

la commande plot peut prendre un troisième paramètre d'entrée qui est une chaîne de 3 caractères **plot(x,y,'cst')** avec '**c'** désignant la couleur du trait, '**s'** le symbole du point et '**t'** le style de trait. On peut voir les différentes possibilités pour ces paramètres en tapant **help plot.** Il n'est pas obligatoire de spécifier chacun des trois caractères.

- **grid on** met la grille sur le graphe tracé par plot. **grid off** efface la grille.
- **title ('le titre'**) permet de donner un titre à la figure.
- **xlabel** et **ylabel** écrivent du texte le long de l'axe correspondant.
- **text(x,y,'texte à afficher')** écrit un texte aux coordonnées du point x,y du graphe.
- **gtext('texte à afficher')** donne un curseur qu'on amène à l'endroit où l'on désire placer le texte. Celui ci s'écrit quand on clique la souris.
- **axis([xmin,xmax,ymin,ymax])** impose les échelles en x et y. s'exécute après la commande plot. **axis('square')** présente le graphe dans un carré au lieu du rectangle habituel.
- **legend** permet d'associer une légende à chaque courbe de la figure.

le tracé de **plusieurs graphes :**

5.3. Gestion des fenêtres graphiques (gestion des figures)

Pour tracer plusieurs graphes dans différentes fenêtres graphiques séparées, on peut taper : **figure(n)** avant la commande **plot**, ou **n** représente le numéro de la fenêtre graphique (le numéro de la figure).

- **close** ferme la figure courante **close(n)** ferme la figure numéro **n**.
- **close all** ferme toutes les figures ouvertes
- **clf** efface la figure courante (en la laissant ouverte).
- **print** permet de sauvegarder la figure d'une fenêtre graphique dans un fichier sous divers formats d'images.

5.4. Affichage de plusieurs courbes dans un seul cadran de la figure

Il y a deux façons de représenter deux courbes sur le même cadran de la figure :

En mettant dans la même commande **plot** plusieurs paires (abscisses, ordonnées)

Exemple :

```
x=0:0.01:2*pi ;
y1=sin(x) ; y2=sin(2*x) ;
plot(x,y1,x,y2)
```

En utilisant la commande « **hold** ». La commande **hold on**, retient le contenu de la fenêtre graphique de façon à pouvoir superposer une nouvelle courbe sur la même fenêtre. La commande **hold off** relâche la fenêtre.

5.5. Affichage de plusieurs courbes dans plusieurs cadrans de la figure :

Il est possible de diviser la figure en plusieurs cadrans dans lesquels on peut mettre différentes courbes grâce à la commande **Subplot(n,m,k),** avec **n** : nombre de lignes, **m** : nombre de colonnes et **k** : sert à spécifier dans quel cadran (sous-fenêtre) doit s'effectuer l'affichage. Les cadrans sont numérotés de gauche à droite et de haut en bas.

L'affichage des vecteurs

L'Affichage des valeurs d'un vecteur :

Pour afficher les valeurs d'un vecteur A = [1 6 3 5 7 9 12], il suffit de taper **plot(A)**. (Essayez cet exemple)

En abscisse, la valeur représentée correspond au numéro de l'indice de façon à pouvoir superposer une nouvelle courbe sur la même fenêtre. La commande **hold off** relâche la fenêtre.

5.6. Affichage de plusieurs courbes dans plusieurs cadrans de la figure :

Il est possible de diviser la figure en plusieurs cadrans dans lesquels on peut mettre différentes courbes grâce à la commande **Subplot(n,m,k),** avec **n** : nombre de lignes, **m** : nombre de colonnes et **k** : sert à spécifier dans quel cadran (sous-fenêtre) doit s'effectuer l'affichage. Les cadrans sont numérotés de gauche à droite et de haut en bas.

5.7. Affichage des valeurs d'un vecteur en fonction d'un autre vecteur :

Si on veut afficher un vecteur A en fonction d'un autre vecteur X, il suffit de taper : **plot(X,A)**.

Matlab interprète le premier vecteur comme celui correspondant à l'axe des abscisses et le second vecteur comme celui correspondant à l'axe des ordonnées.

REMARQUE :

D'Autres fonctions sont intéressantes pour la représentation des figures. On a : **plot3**, **polar, bar, hist, quiver, compass, feather, rose, stairs, fill**. Utilisez la fonction **help** pour obtenir une description de ces commandes.

5.7. Graphiques 2D

TRAÇAGE DE COURBES

On utilise l'instruction plot pour tracer un graphique 2D:
plot(x,y) Tracer le vecteur y en fonction du vecteur x
plot(t,x,t,y,t,z) Tracer x(t), y(t) et z(t) sur le même graphique
plot(t,z,'r--') Tracer z(t) en trait pointillé rouge

FORMAT DE GRAPHIQUE

On peut choisir le format du graphique:

- plot(x,y) Tracer y(x) avec échelles linéaires
- semilogx(f,A) Tracer A(f) avec échelle log(f)
- semilogy(w,B) Tracer B(w) avec échelle log(B)
- polar(theta,r) Tracer r(theta) en coordonnées polaires
- bar(x,y) Tracer y(x) sous forme des barres
- grid Ajouter une grille

Exemple:

```
>> t=0:0.01e-3:0.06;
>> y=10*exp(-60*t).*cos(120*pi*t);
>> z=10*exp(-60*t).*sin(120*pi*t);
>> plot(t,y,'r',t,z,'g'),grid
>> a=10*exp(-60*t);
>> hold
Current plot held
>> plot(t,a,'b--')
>> plot(t,-a,'b--')
>> title('Fonctions sinusoidales amorties')
>> xlabel('Temps , s'),ylabel('Tension , V')
>> hold off
>> plot(y,z),grid
>> axis equal
>> xlabel('y'),ylabel('z')
```

5.8.GRAPHIQUE MULTIPLE

On peut tracer plusieurs graphiques dans la même fenêtre en utilisant l'instruction subplot pour diviser la fenêtre en plusieurs parties.

- Diviser la fenêtre en deux parties (2 x 1)

Exemple:

```
>> w=logspace(0,3,1000);
>> s=j*w;
>> H=225./(s.*s+3*s+225);
>> AdB=20*log10(abs(H));
>> phase=angle(H)*(180/pi);
>> subplot(2,1,1),semilogx(w,AdB),grid
>> xlabel('w , rad/s'),ylabel('Amplitude , dB')
>> subplot(2,1,2),semilogx(w,phase),grid
>> xlabel('w , rad/s'),ylabel('Phase , degre')
subplot(2,1,1)
```

```
subplot(2,1,2)
```

- Diviser la fenêtre en deux parties (1 x 2)
- Diviser la fenêtre en quatre parties (2 x 2)
- Diviser la fenêtre en quatre parties (4 x 1)

5.8. AJOUT DU TEXTE AU GRAPHIQUE

title('Titre du graphique') Donner un titre au graphique

xlabel('Temps') Étiquette de l'axe x

ylabel('Tension') Étiquette de l'axe y

gtext('Valeur absolue') Ajouter du texte au graphique avec la souris

subplot(1,2,1) subplot(1,2,2)

subplot(2,2,1) subplot(2,2,2)

subplot(subplot(2,2,3) 2,2,4)

subplot(4,1,1)

subplot(4,1,2)

subplot(4,1,4)

subplot(4,1,3)

MANIPULATION DE GRAPHIQUES

axis([-1 5 -10 10]) Choix des échelles x = (-1,5) et y = (-10,10)

hold Garder le graphique sur l'écran (pour tracer plusieurs courbes sur le même graphique)

IMPRESSION ET ENREGISTREMENT DE GRAPHIQUES

print -dps Imprimer le graphique en PostScript

print -dpsc Imprimer le graphique en PostScript Couleur

print -dps dessin.ps Enregistrer le graphique en PostScript dans le fichier dessin.ps

5.8. Graphiques 3D

Le traçage des graphiques 3D est illustré dans les deux exemples suivants.

Exemple:

```
>> t = 0:0.05:25;
>> x = exp(-0.05*t).*cos(t);
>> y = exp(-0.05*t).*sin(t);
>> z = t;
```

```
>> plot3(x,y,z), grid
```

6. Programmation avec Matlab

COMMUNICATION AVEC L'USAGER

On peut afficher un message, une valeur à l'écran avec l'instruction disp:

disp('Ceci est un test') Afficher "Ceci est un test" sur l'écran

On peut entrer une valeur avec l'instruction input:

x = input('Valeur de x = ') Afficher sur l'écran "Valeur de x = " et attendre qu'un nombre soit tapé sur le clavier

BOUCLE FOR

On peut créer une boucle en utilisant for ... end. On peut aussi réaliser des boucles FOR imbriquées.

Exemple:

```
%Boucle FOR simple:
for i=1:100
wt = 24*i*0.01;
x(i)=12.5*cos(wt+pi/6);
end
```

Deux boucles FOR:

```
for i=1:5
for j=1:20
amp=i*1.2;
wt=j*0.05;
v(i,j)=amp*sin(wt);
```

```
end

end
```

BOUCLE WHILE

On peut créer une boucle en utilisant while ... end.

Exemple:

```
n=1;

while n<100

x=n*0.05;

y(n)=5.75*cos(x);

z(n)=-3.4*sin(x);

n=n+1;

end
```

INSTRUCTION IF ... ELSEIF ... ELSE

L'instruction IF ... ELSEIF ... ELSE permet de choisir plusieurs options.

Exemple 11:

```
n=input('Donner un nombre positif ');

if rem(n,3)==0

disp('Ce nombre est divisible par 3')

elseif rem(n,5)==0

disp('Ce nombre est divisible par 5')

else

disp('Ce nombre n''est pas divisible par 3 ou par 5')
```

```
end
```

7. FICHIERS M

Les fichiers M sont des fichiers ASCII contenant des suites d'instructions MATLAB dont le nom a comme extension m. Par exemple «test1.m». Dans la fenêtre Commande, si l'on tape

test1, les instructions contenues dans le fichier test1.m seront exécutées une par une. On peut créer des fichiers M à l'aide de «Text Editor».

Exemple d'un fichier M:

```
% Ceci est un exemple de fichier M
% Les lignes "commentaires" commencent par "%"
for i=1:10
for j=1:4
x=0.005*i;
y=30+j;
z(i,j)=10*exp(-y*x)*cos(120*pi*x);
end
end
```

CRÉATION DE FONCTIONS MATLAB

Une fonction MATLAB est un fichier M particulier dont la première ligne commence par

«function». Une fonction peut être utilisée dans les expressions mathématiques ou dans les instructions

MATLAB.

Exemple d'une fonction MATLAB:

```
function y = EFF(x)
% EFF Calcul de la valeur efficace
```

```
% Pour un vecteur EFF(x) donne la valeur efficace
% Pour une matrice, EFF(x) donne un vecteur contenant
% la valeur efficace de chaque colonne.
[m,n] = size(x);
if m==1
m=n;
end
y=sqrt(sum(x.*x)/m);
```

Les commentaires donnés dans la fonction EFF seront affichés à l'écran lorsqu'on tape help EFF.

Travail demandé

1-

On cherche à **résoudre une équation du second degré à variable réelle x de type**

$$ax^2 + bx + c$$

Ecrire le programme qui donne la solution de l'équation du second degré à variable réelle x de type $ax^2 + bx + c$

2-

On a besoin de tracer la courbe de la fonction :

$y = \sin(x)$ Pour $x \in [0, 2\pi]$, on écrit alors :

Ecrire le programme qui permit de tracer cette courbe

3-

On considère la fonction $y(t) = 4\exp\left(-\frac{(t-5)^2}{2}\right)$, écrire un script Matlab permettant de :

1) créer un vecteur t de 150 points, de valeurs comprises entre 0 et 10.
2) Tracer la courbe correspondant à la fonction (t)
3) Fixer la taille de la fenêtre graphique de façon à ce que les abscisses soient comprises entre -1 et 11 et les ordonnées entre -2 et 5.
4) Compléter la figure en y ajoutant les axes et un titre.

4-

Soit la fonction $(x) = x^2 - 1$ prise sur un intervalle $= [-5,5]$. Écrire un script Matlab qui permet de :

1) Définir l'intervalle de la variable x
2) Tracer la fonction (x).

3) Tracer la fonction $f'(x)$ (la dérivée de $f(x)$). 4) Attribuer un titre aux deux graphes.

Solution du travail demandé

1-

Code matlab

```
disp('Programme qui résoud une équation du second degré à varibale réelle x de type ax^2+bx+c avec')
a=input('a=');
b=input('b=');
c=input('c=');
delta=b^2-4*a*c;
if delta<0
   disp('pas de solutions dans R');
elseif delta==0
   x=-b/(2*a);
else
   x1=(-b+sqrt(delta))/(2*a);
   x2=(-b-sqrt(delta))/(2*a);
end
if delta==0
   x
elseif delta>0
   x1,x2
end
```

2-

```
x=0:0.01:2*pi  % car on a besoin de déclarer le vecteur des abscisses x

y=sin(x)

 % pour évaluer le vecteur des ordonnées y

plot(x,y)      % pour tracer y en fonction de x (y=f(x))
```

3-

Code Matlab :

```
%Exemple
clear all    %effacer les variables en cours
close all    %fermer toutes les figures ouvertes
clc          %effacer l'écran
t=linspace(0,10,150);    % car on a besoin de déclarer le vecteur des abscisses t
y=4*exp(-((t-5).^2)./2);
plot(t,y);grid
%axis([-1 11 -2 5])
xlabel('le temps t')
ylabel ('la fonction y(t)')
title('le signal de sortie y(t) ')
```

4-

la dérivé d'une fonction $f(x)$ est donnée par la formule $f'(x) = \frac{df}{dx}$. Sur Matlab on peut calculer df et dx par l'instruction « **diff** ». La fonction **diff** appliquée à un vecteur **x** de taille **n** donne un vecteur **dx** de taille (**n-1**) dont chaque élément correspond à la différence de deux éléments successifs de **x**.

Code Matlab :

```
%Exemple
clear all    %effacer les variables en cours
close all    %fermer toutes les figures ouvertes
clc          %effacer l'écran
x=-5:0.001:5 ;  % car on a besoin de déclarer le vecteur des abscisses t
f=x.^2-1   % évaluer la fonction f(x)
dx=diff(x);
df=diff(f);
fprim=df./dx   % évaluer la dérivé de la fonction f(x)
% le tracer les deux côte à côte dans la même figure
subplot(2,1,1), plot(x,f), grid, title('la fonction f(x)'),
subplot(2,1,2), plot(x(1:end-1),fprim), grid, title('la dérivée de f(x)')
```

TP 1 :

Représentation de signaux et applications de la transformée de Fourier sous Matlab

1- Objectif du TP :

L'objectif de cette partie est de générer, de visualiser quelques signaux analogiques, de déterminer leur produits de convolutions et leur corrélation en utilisant MATLAB.

A la fin de la séance de travaux pratiques l'étudiant doit être capable de :

- Générer des signaux continus.
- Réaliser la convolution entre deux signaux analogiques.

2-Rappel théorique

Un signal est la représentation physique de l'information. La description mathématique des signaux est l'objectif de la théorie du signal. Elle offre les moyens d'analyser, de concevoir et de caractériser les systèmes de traitement de l'information.

2-1-Représentation temporelle des signaux

Cette représentation est basée sur l'évolution du signal en fonction du temps. On distingue deux types fondamentaux de signaux :

2.1.1. ***Les signaux certains ou déterministes*** Leur évolution en fonction du temps peut être parfaitement décrite par un modèle mathématique. Parmi les signaux déterministes on distingue :

****Les signaux périodiques*** : ce sont les signaux dont l'évolution dans le temps est prévisible et qui obéissent à une loi de répétition cyclique régulière, de période T.

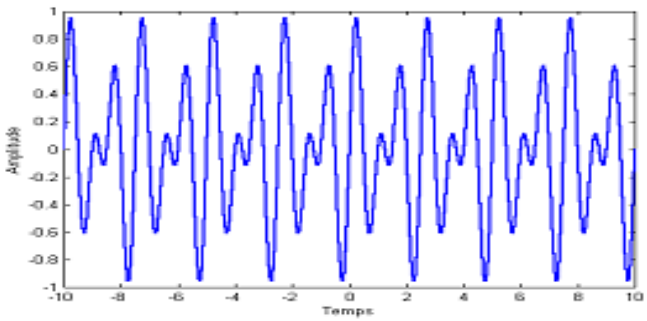


Figure 1 : signal périodique S(t)°=S(T+K), k est un entier

Les signaux sinusoïdaux sont un cas particulier de ces signaux :

$$S(t)=A\sin[(2.\pi/T)t+\varphi]$$

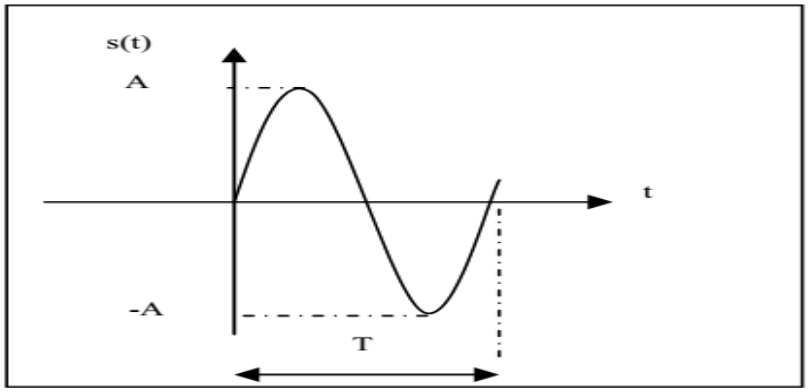


Figure 2 : signal sinusoïdal

2.1.2. *Les signaux aléatoires* :

Ce sont les signaux dont le comportement temporel est imprévisible, gouvernée par les lois du hasard.

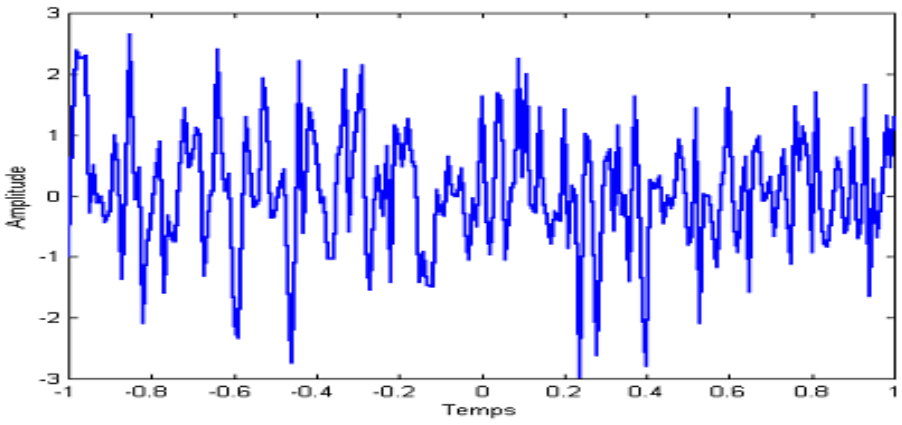


Figure 3 : signal aléatoire

2 .2. Produit de Convolution

2.2.1. Définition du produit de convolution

En traitement du signal, la convolution est l'outil permettant de calculer la sortie d'un système. En effet, pour un signal d'entrée e(t) soumis a un système de fonction transfert h(t), la sortie sera la convolution des deux fonctions $h(t) * e(t)$.

La convolution de e(t) par e par h (t). et noté « * ».

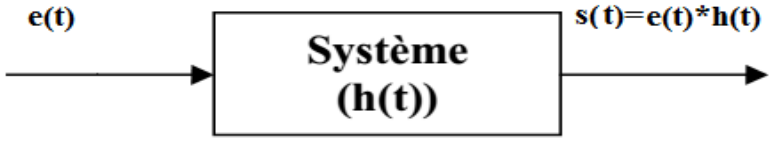


Figure 4. Réponse du système

2.2.2. Equation générale de convolution :

$$s(t) = e(t) * h(t) = \int_{-\infty}^{+\infty} e(t-\tau)h(\tau)d\tau = \int_{-\infty}^{+\infty} e(\tau)h(t-\tau)d\tau \; produit\ de\ convolution$$

3. Simulation par MATLAB

3.1. Représentation de quelques signaux par MATLAB

Soit quelques notions de base de Matlab que vous allez utiliser :

Plot : Trace une représentation graphique.

Grid : affiche une grille.

Title : attribue un titre au graphique.

Xlabel : attribue un texte à l'axe des abscisses.

Ylabel : attribue un texte à l'axe des ordonnées.

pi : c'est la valeur 3.14

2. Transformée de Fourier continue sous Matlab (Signaux analogiques)

L'objectif de cette partie est de réaliser la Transformée de Fourier et d'étudier le principe d'échantillonnage des signaux analogiques par simulation à l'aide du logiciel MATLAB.

Le but aussi est de déterminer leur transformée de Fourier discrète TFD et rapide FFT en utilisant **MATLAB**.

2-1-Rappel théorique

Un signal est la représentation physique de l'information. La description mathématique des signaux est l'objectif de la théorie du signal. Elle offre les moyens d'analyser, de concevoir et caractériser des systèmes de traitement de l'information.

Représentation fréquentielle des signaux

2.1. 1. Introduction

La transformation de Fourier, généralisée par l'emploi des distributions, permet d'obtenir une représentation spectrale des signaux déterministes. Celle-ci exprime la répartition fréquentielle de l'amplitude, de phase, de l'énergie ou de puissance des signaux considérés.

2.1.2. Transformée de Fourier

Définition de la TF:

Soit x(t) un signal déterministe, sa Transformée de Fourier est une fonction généralement complexe, de la variable réelle f définis par :

$$X(f) = \int_{-\infty}^{+\infty} x(t).e^{-j2\pi ft}.dt$$

Travail demandé

Ce premier TP est une initiation au traitement du signal sous Matlab. Son objectif est d'apprendre à générer, visualiser et analyser quelques signaux analogiques et calculer la transformée de Fourier ainsi que de mettre en application les connaissances acquises sur la transformée de Fourier.

1- donner les représentations des signaux suivants sous Matlab ?

- L'impulsion de Dirac δ(t)
- Génération d'une impulsion rectangulaire
- Génération du sinus cardinal sinc (x)
- Représentation de signal carré
- Représentation de signal triangulaire
- Représentation de signal dents
- Représentation de signal sinusoïdal

2-

Ecrire un programme chargé d'engendrer et tracer sur la même figure, les signaux de base

1-sur l'intervalle [0 :N-1]avec N=20

-sinusoïde de fréquence 50hz (prendre d'autres valeurs de fréquence pour voir)

-impulsion Unité(utiliser **eys(1,N**))

-échelon unité(**utiliser ones(1,N))**

2- une porte de 2*p+1 points avec p=3 sur l'intervalle [-N,N]

Utilisez **zeros(1,N-p) et ones(1,2*p+1))**

3-

Dans un même programme :

Générer 128 points de la séquence : $x(n)=A\sin(\pi n/12)$ avec A=5

Utilisation de **randn** pour générer 128 point de la séquence r(n), en suite on a tracer dans la même figure x(n) et s(n)=x(n)+r(n) :

4. Produit de convolution de deux signaux analogiques

Donner le programme Matlab qui calcule le produit de convolution (La commande : ***conv***) de deux signaux rectangulaires , l'un de durée 20 s et d'amplitude 2v et l'autre de durée 40s et d'amplitude 3v.

5-Transformée de Fourier d'un signal rectangulaire

Donner le programme **Matlab** suivant qui calcule-la transformée de Fourrier d'un signal rectangulaire centré, d'amplitude A=1v et de largeur T= 20s.

6-Transformée de Fourier d'un signal cosinus :

Donner le programme qui permet de construire et d'afficher un signal cosinus x(t) d'amplitude 1v, de fréquence f_0. Ce vecteur x est composé de N points et représente r périodes du cosinus. La variable temps du signal est entre 0 et Tmax = r/f0.

Le programme permet aussi de calculer et d'afficher la transformée de Fourier de ce signal en utilisant la définition de la transformée de Fourier et aussi en utilisant la commande fft (Fast Fourier Transform) de Matlab (Voir le help de fft, abs et fftshift).

Solution
du Travail demandé

1-

- **Soit le programme suivant de *l'impulsion de Dirac δ(t)***

```
%Génération d'une impulsion unité
%Génération d'un vecteur de -10 0 20
n=-10:20;
%Génération de l'impulsion unité
u=[zeros(1,10) 1 zeros(1,20)];
%Tracer le signal généré
stem(n,u);
xlabel('Temps indexé en n');
ylabel('Amplitude');
title('impulsion unité');
axis([-10 20 0 1.2]);
```

- ***Génération d'une impulsion rectangulaire***

  Soit le programme suivant :

```
t = -1:0.00001:1 ;
x1 = rectpuls(t,0.05) ;
plot(t,x1) ; axis([-0.1 0.1 -0.2 1.2])
; grid ;
xlabel('Temps(sec)');
ylabel('Amplitude');
 title('impulsion rectangulaire ') ;
```

- ***Génération du sinus cardinal sinc (x)***

Soit la fonction $y(x) = sinc(x) = \sin(x)/x$

On utilise l'expression logique (x= =0) pour exprimer : que la $\lim_{x\to 0} y(x) = 1$

Soit le programme suivant :

```
%Tracage de la fonction sinus cardinal
%Domaine des valeurs de la variable x
x=-4*pi:pi/100:4*pi;
%valeurs de la fonction
y=(x==0)+sin(x)./(x+(x==0));
%Tracage de la fonction sinus cardinal
plot(x,y)
grid
title('sinus cardinal y=sin(x)/x')
```

- ***Représentation de signal carré :***

Soit les programmes ci- dessous :

```
fs = 10000 ;
t = 0:1/fs:1.5 ;
y = square(2*pi*50*t) ;
plot(t,y), axis([0 0.1 -1.2
1.2]),
 grid
xlabel('Temps (sec)') ;
ylabel('Amplitude') ;
title('signal ……..')
```

- ***Représentation de signal triangulaire :***

Soit les programmes ci- dessous :

Programme :

```
t = -1:0.00001:1 ;
x2 = tripuls(t,0.04) ;
plot(t,x2),
axis([-0.1 0.1 -0.2 1.2]),
grid ;
xlabel('Temps (sec)') ;
ylabel('Amplitude') ;
 title('impulsion ……') ;
```

- ***Représentation de signal dents :***

Soit les programmes ci- dessous :

Programme :

```
fs = 10000 ;
t = 0:1/fs:1.5 ;
y1 = sawtooth(2*pi*50*t) ;
plot(t,y1); axis([0 0.1 -1.2
1.2]); grid;
xlabel('Temps (sec)') ;
ylabel('Amplitude') ;
 title('signal ..... ') ;
```

- ***Représentation de signal sinusoïdal***

Soit les programmes ci- dessous :

Programme :

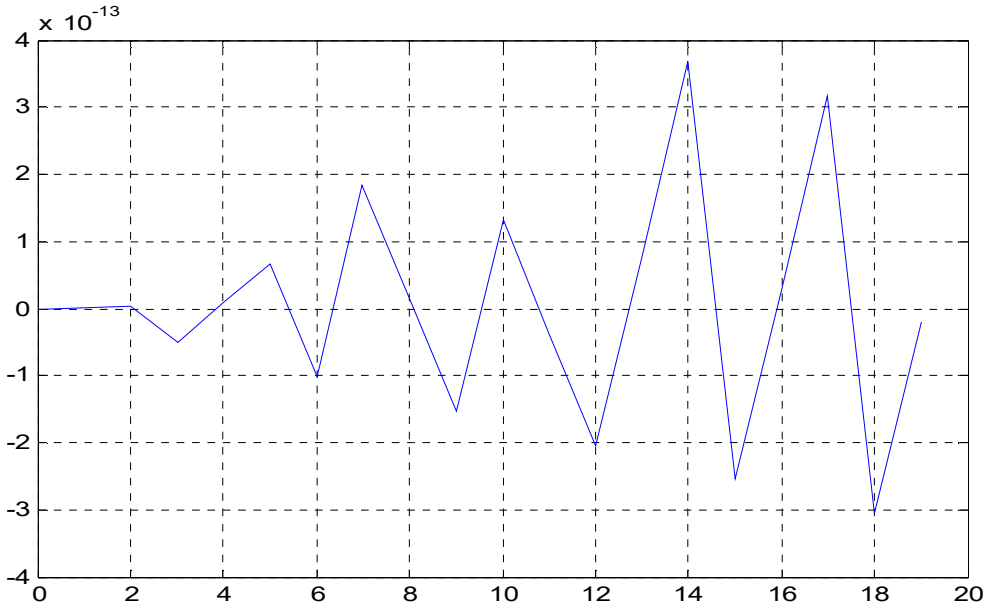
```
x=0:0.001:4*pi ;
y=sin(x) ;
plot(x, y) ;
axis([0 4*pi -1.2 1.2]) ;
grid ;
```

2-

Sur l'intervalle [1 :N-1] avec N=20 :

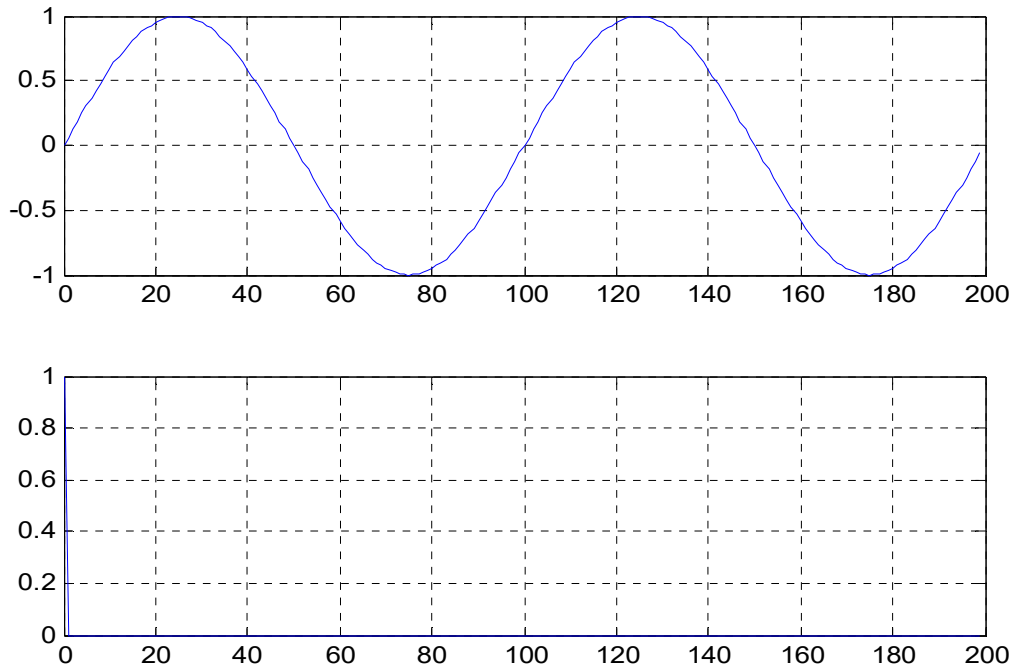
Sur Matlab :

```
N=20;
f= 50; %valeur de la fréquence f = 50 Hz
t=[0:N-1];
fsin=sin(2*pi*f*t); %sinusoide
plot(t,fsin) , grid
```



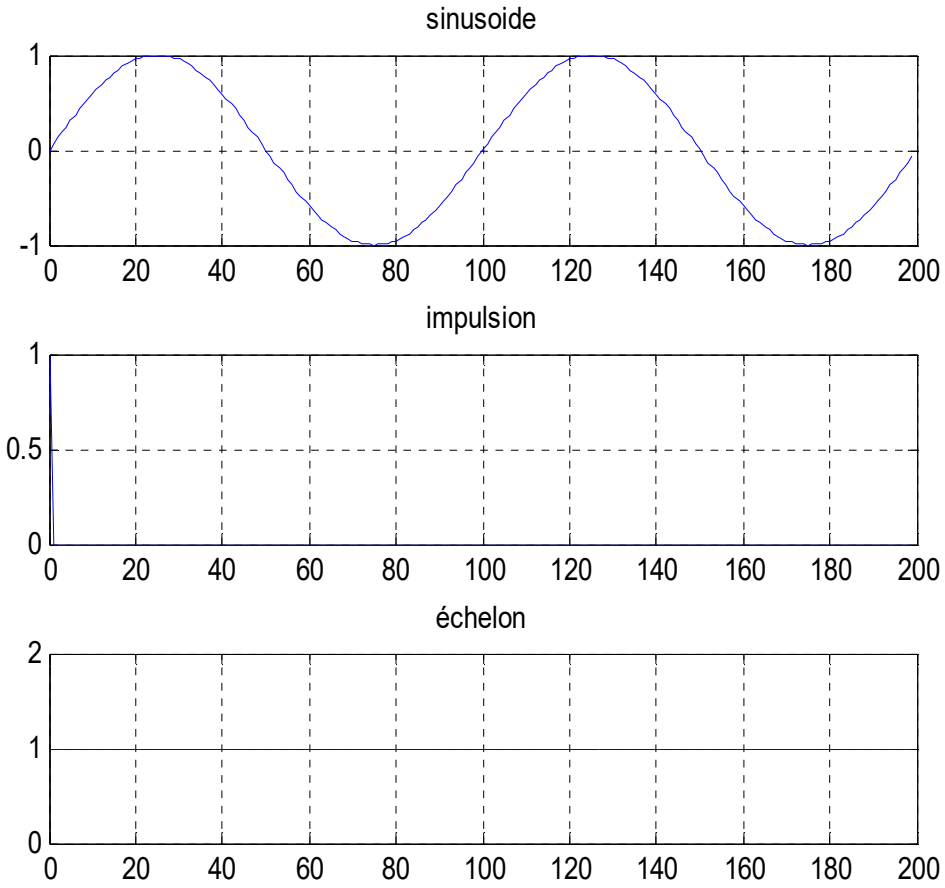
En suite :

```
N=200;
f= 0.01; %valeur de la fréquence f = 0.01 Hz
t=[0:N-1];
fsin=sin(2*pi*f*t); %sinusoide
impl=eye(1,N); %impulsion unité
subplot(2,1,1); plot(t,fsin) , grid
subplot(2,1,2) ; plot(t,impl) , grid
```



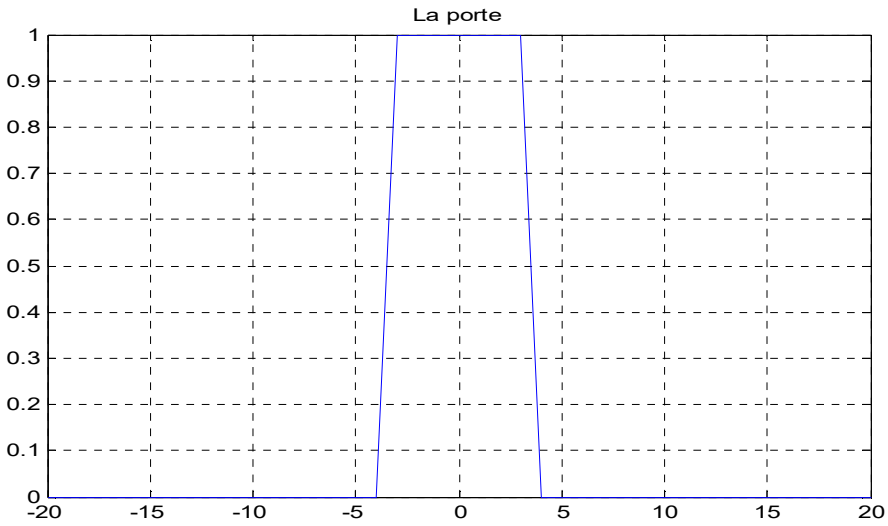
Et on ajoute l'échelon :

```
N=200;
f= 0.01; %valeur de la fréquence f = 0.01 Hz
t=[0:N-1];
fsin=sin(2*pi*f*t); %sinusoide
impl=eye(1,N); %impulsion unité
echel=ones(1,N); %echelon unité
subplot(3,1,1); plot(t,fsin) , grid
title('sinusoide');
subplot(3,1,2) ; plot(t,impl) , grid
title('impulsion');
subplot(3,1,3) ; plot(t,echel) , grid
title('échelon');
```



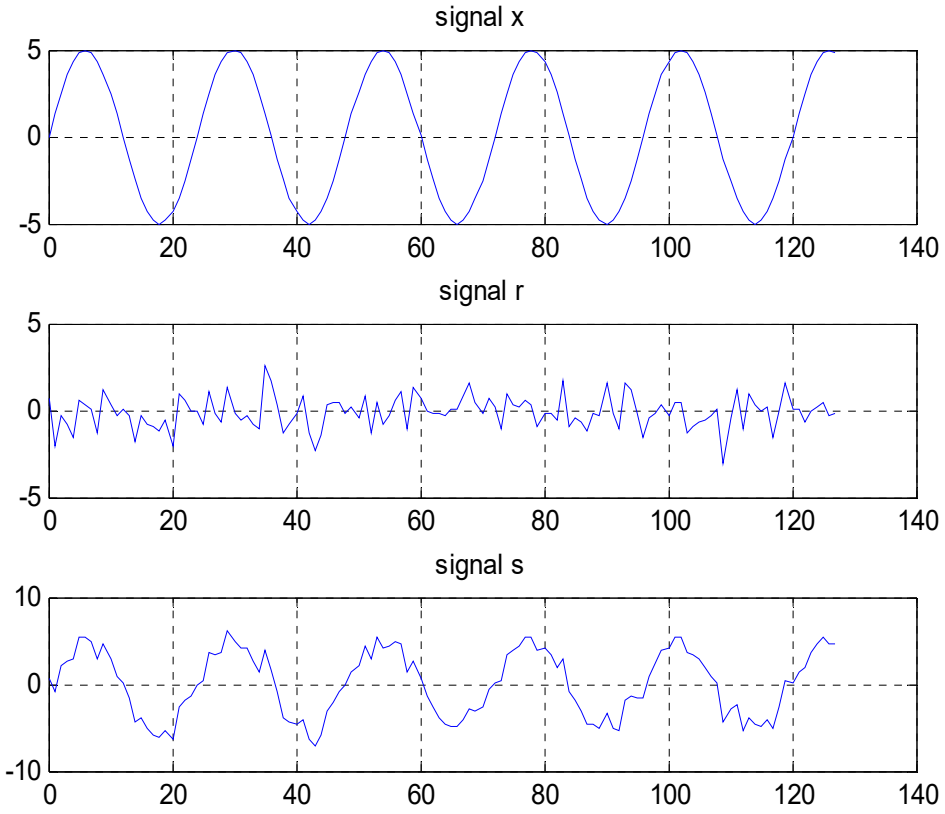
Une porte de 2*P+1 point avec P=3 sur l'intervalle [-N :N] :

```
P=3;
N2=20;
t2=[-N2:N2]; %l'intervalle 2
Portep=[zeros(1,N2-P),ones(1,2*P+1),zeros(1,N2-P)];
plot(t2,Portep), grid
title('La porte')
```

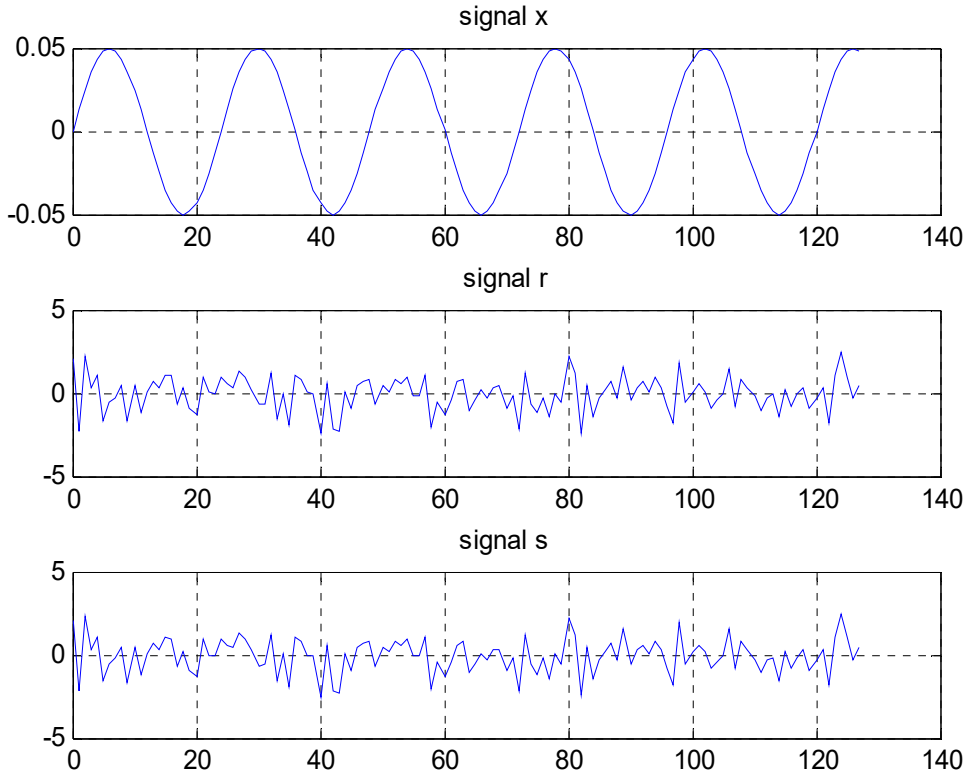


3-

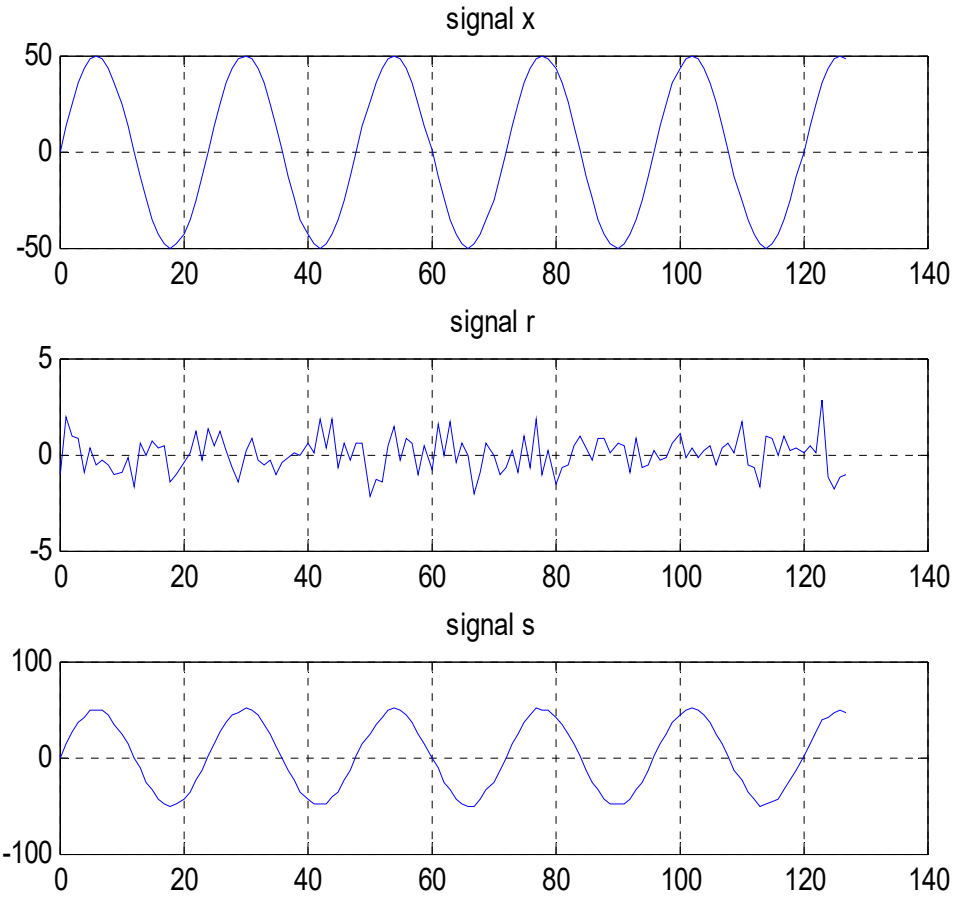
```
A=5;
N=128;
n=[0:1:N-1];
x=A*sin((pi*n)/12);
r=randn(1,N);
s=x+r;
subplot(3,1,1);plot(n,x), grid
title('signal x')
subplot(3,1,2);plot(n,r) , grid
title('signal r')
subplot(3,1,3);plot(n,s) , grid
title('signal s')
```



Pour A=0.05 :



Pour A=50 :



4-

Programme :

```
clear
x=zeros(100,1);
for i=41:60,x(i)=2;end;
y=zeros(100,1);
for i=31:70,y(i)=3;end;
N=100;
a=-N/2:N/2-1;
b=-N:N-2;
subplot(2,2,1);plot(a,x);axis([-50,50,0,2.5]);grid;
title('premier signal');xlabel('t');ylabel('x(t)');
subplot(2,2,2);plot(a,y);axis([-50,50,0,3.5]);grid;
title('deuxième signal');xlabel('t');ylabel('y(t)');
subplot(2,2,4);plot(b,conv(x,y));grid;
axis([-60 60 0 140]);title('convolution');xlabel('t');
```

5-

- **Soit le Programme suivant :**

```
clear;N=400;x=zeros(N,1);     % N= nombre de points
T=20;                         %largeur du signal rect
for i=N/2-T/2:N/2+T/2-1,x(i)=1;end;
t=-N/2:N/2-1;
subplot(211);plot(t,x);axis([-N/2,N/2,0,1.3]);
grid;xlabel('t(s)');
f=-0.5:1/N:0.5-1/N;g=fft(x,N);
subplot(212); plot(f,fftshift(abs(g(1:N))),'-r');
xlabel('F(Hz)');grid
```

6-

Soit le Programme suivant :

```
clear ; f0=10; N=1000 ;
r=3; Tmax=r/f0;
t=0:Tmax/N:Tmax-Tmax/N;
x=cos(2*pi*f0*t);
subplot(2,2,1);plot(x);grid;
g=fft(x,N)/N;subplot(2,2,2);
f=(-N/2:N/2-1)/r*f0;
plot(f,fftshift(abs(g)));
axis([-2*f0,2*f0,-0.1,0.6]); grid;
title('Module de la TF par FFT');
xlabel('F(Hz)'); ylabel('amplitude');
i=1:N;d=[];
for u=-0.5:1/N:0.5-1/N;
r=sum(x.*cos(2*pi*u*i))/N;
im=sum(x.*sin(2*pi*u*i))/N;
d=[d norm([r im])];end;
subplot(2,2,3); plot(f,d);
axis([-2*f0,2*f0,-0.1,0.6]);
grid;title('Module de la TF par la définition');
xlabel('F(Hz)');ylabel('amplitude');
```

Durant ce premier TP nous avons fait un rappel, sur les différents signaux l'échelon, la rampe, Sinusoïdale et sur produit de convolution de deux signaux analogiques et calcule de transformée de Fourier.

TP 2 :

Filtrage Analogique

Objectif du TP :

L'objectif de ce TP est de mettre en application les connaissances acquises sur le filtrage .Nous verrons, grâce à Matlab, les différents filtres idéals appliqué dans divers domaines tels que l'électronique, l'automatique…..

1-Partie theorique

Synthese d'un filtre analogique

1-1- Filtre passe bas :

Un filtre passe bas permet d'attenuer les frequences superieurs à la fréquence de coupure choisie et laisse passer les basse fréquences

Shcema d'un filtre passe bas passif :

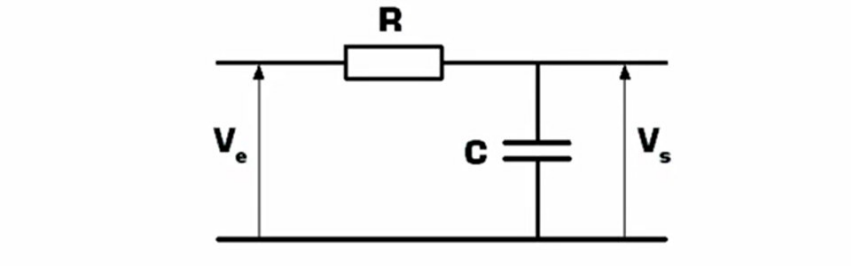


Figure 2-1 filtre passe bas

Nous allons determiner sa fonction de transfert **H(jw)**

$$\mathbf{H(jw)} = \frac{\frac{1}{jCW}}{R+\frac{1}{jCW}} = \frac{1}{jCW} * \frac{jCW}{1+jRCW} = \frac{1}{1+jRCW}$$

Or $\tau = RC$ et on sait que : $Wc = \frac{1}{\tau}$

D'où $\mathbf{H(jw)} = \frac{1}{1+j\frac{W}{Wc}}$ et en posant $x = \frac{W}{Wc}$ on obtient :

$$\mathbf{H(jw)} = \frac{1}{1+jX}$$

Notre objectif est de programmé cette fonction sur **MATLAB**

1-2- Filtre passe haut :

Un filtre passe haut permet d'attenuer les frequences inferieurs à la fréquence de coupure choisie et et concerver les hautes frequences.

Shcema d'un filtre passe haut passif :

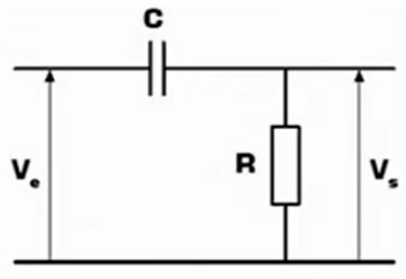


Figure 2.2 filtre passe haut

Nous allons determiner sa fonction de transfert **H(jw)**

$$H(jw) = \frac{R}{R+\frac{1}{jCW}} = R * \frac{jCW}{1+jRCW} = \frac{jRCW}{1+jRCW}$$

Or $\tau = RC$ et on sait que : $Wc = \frac{1}{\tau}$

D'où $H(jw) = \frac{j\frac{W}{Wc}}{1+j\frac{W}{Wc}}$ et en posant $x = \frac{W}{Wc}$ on obtient :

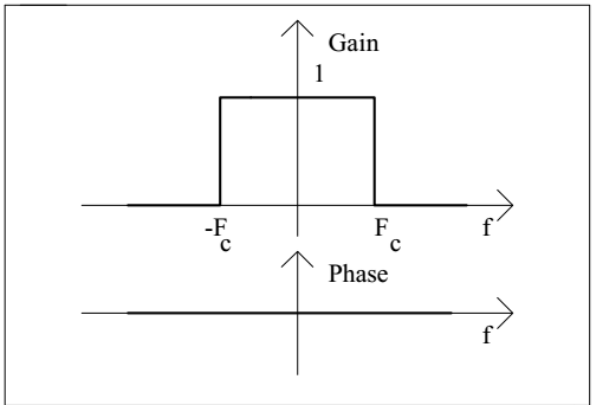
$$H(jX) = \frac{jX}{1+jX}$$

Notre objectif est de programmé cette fonction sur **MATLAB**

2-3-Le filtre passe bande :

Dans cette partie, on étudie le filtrage idéal d'un signal porte.

$$\Pi_T(t) = \begin{cases} 1 & pour\ -T/2 < t < T/2 \\ 0 & ailleurs \end{cases}$$



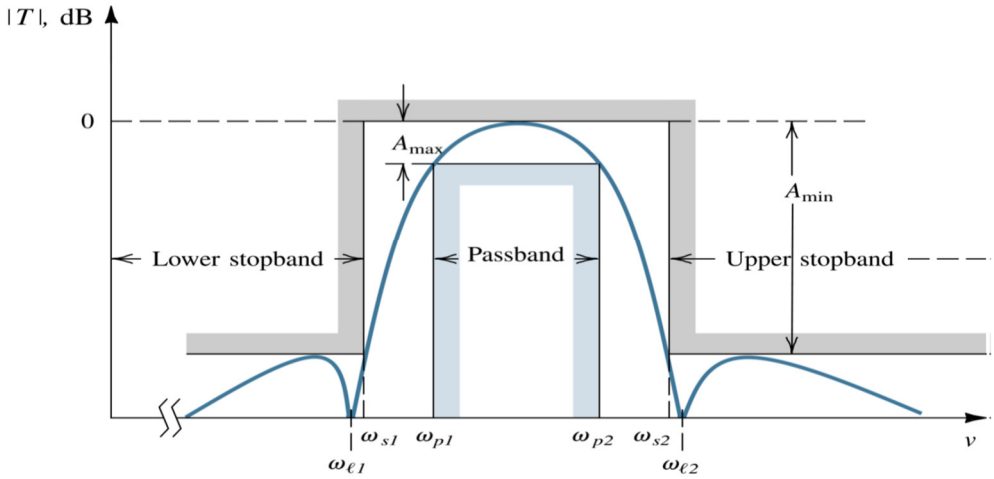


Figure 2.3 :filtre passe_bande idéal

On considère un filtre H(f) passe bande idéal de fréquence de coupure $f_c = 2/T$

```
%Définitions préalables

T=5;

Fe=100;

Te=1/Fe;

fc=2/T;
```

Définition du domaine temporel et tracé de la fonction porte pour T=5

Pour cela, on crée un vecteur t = [-20…20]. On choisit un pas $T_e=1/F_e=1/100$.

```
%Definition du domaine temporel

t=-20:Te:20;
```

Le domaine temporel étant défini, on peut maintenant tracer la fonction porte pour T=5 :

```
%Creation de la fonction porte
x= (((-T/2) <=t)&(t<=(T/2))) ;
%Representation temporelle de la fonction porte avec T=5
figure(1)
plot(t,x,'r');
grid;
axis([-5 5 -0.05 1.05]);
title('Representation temporelle de la fonction porte avec T=5')
xlabel('temps t')
ylabel('amplitude du signal')
```

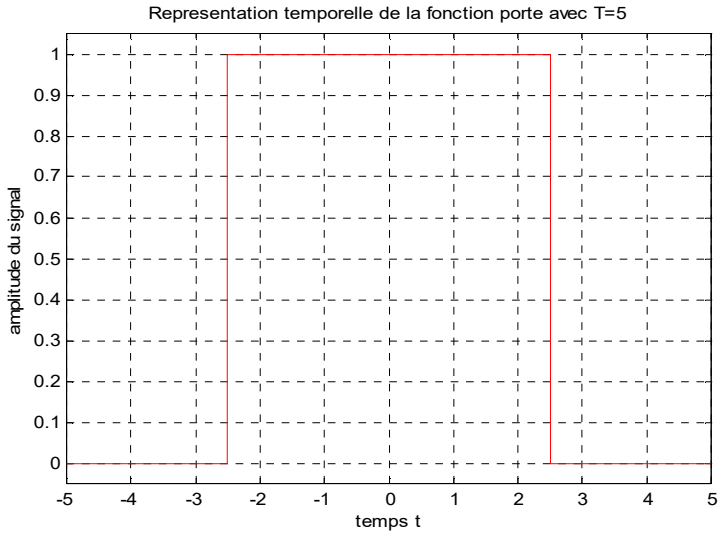


Figure 2.4:représentation temporelle de la fonction porte

Définition du domaine fréquentiel et tracé de la fonction H(f)

Dans le domaine fréquentiel, il faut aussi représenter f sous forme d'un vecteur :

On discrétise donc le domaine fréquentiel. En effet, le calculateur, en l'occurrence Matlab, ne peut calculer le contenu fréquentiel du signal qu'en un nombre fini de points fréquentiels. Pour pouvoir passer du domaine temporel au domaine fréquentiel sans problème, la dimension de f doit être égale à la dimension de t.

On a donc $f = (F_e/2) + k * \Delta(f)$ avec $\Delta(f) = ((F_e/2 - (-F_e/2))/4000$

$= F_e/4000 = 1/40HZ$

```
%Définition du domaine fréquentiel
```

```
f=linspace(-Fe/2,Fe/2,length(t));
```

Le domaine fréquentiel étant défini, on peut maintenant tracer H(f).

H(f) est un passe-bande idéal de fréquence de coupure f_c=2/T=2/5=0.4HZ

On a donc :

$$H(f)=\begin{cases}1 & pour\ -T/2<f<T/2\\ 0 & ailleurs\end{cases}$$

```
%Création du filtre passe bande de fréquence de coupure fc=2/T

Hf=(((-2/T)<=f)&(f<=(2/T)));

%Représentation fréquentielle du filtre passe bande idéal H(f)

figure(2)

plot(f,Hf)

grid

axis([-2 2 -0.05 1.05])

title('Représentation fréquentielle du filtre passe bande idéal H(f) de fréquence de coupure fc=2/T')

xlabel('Fréquence f')

ylabel('amplitude de H(f)')
```

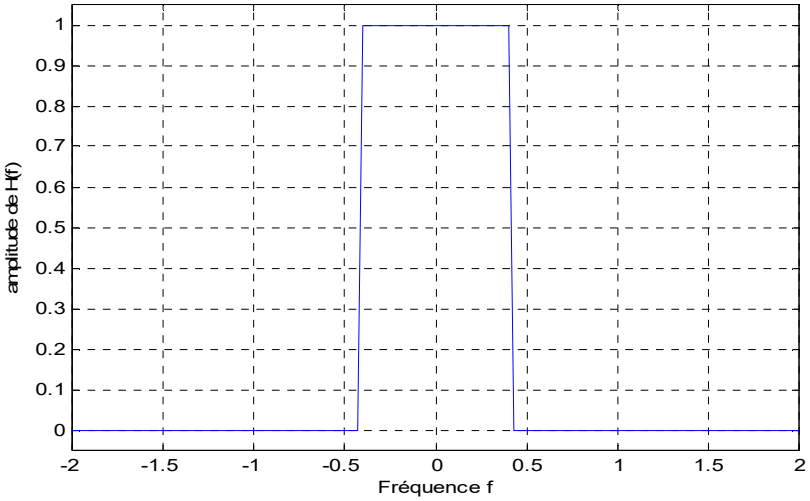


Figure 2.5. Représentation fréquentielle du filtre passe bande idéal h(f)

Calcul et tracé de X(f) entre -2 et 2 HZ

On considère X(f) la transformée de Fourier de la fonction porte $\prod_5(t)$.Pour cela, on utilise la commande fft (transformée de Fourier rapide).

```
%Transformée de Fourier de la fonction porte (appelée x):X(f)=TF[x(t)]
Xf=fftshift(fft(x)*Te);
%Représentation graphique de X(f)
figure(3)
plot(f,abs(Xf),'r')
grid
axis([-2 2 0 5])
title('Représentation fréquentielle du signal x(t): X(f)=TF[x(t)]')
xlabel('Fréquence f')
ylabel('amplitude de X(f)')
```

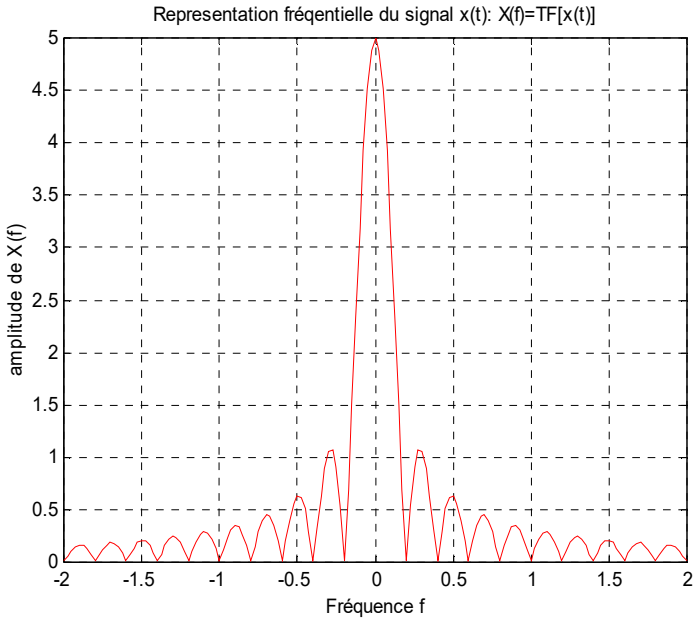


Figure 2.6 : représentation fréquentielle du signal porte

Remarque :

La transformée de Fourier de la fonction porte est un sinus cardinal (sinc), cette fonction est caractérisée par un lobe principal et des lobes secondaires qui s'atténuent.

$TF[\Pi_5(t)]=5[\sin(5\pi f)]/5\pi f=5\text{sinc}(5f)$

Calcul et tracé de Y(f) entre -2 et 2 HZ

y est la sortie du filtre en réponse à l'entrée porte x(t). Soit y(t)=h(t)*x(t). D'après le théorème de Plancherel, on obtient : Y(f)=H(f) . X(f)

$$Y(f)=\begin{cases} X(f) & pour\ -T/2 < f < T/2 \\ 0 & ailleurs \end{cases}$$

```
%Calcul de Y(f)=H(f)X(f)

Yf=Hf.*Xf;

%Représentation fréquentielle

figure(4)

%A) Représentation graphique de H(f) et X(f)

subplot(2,1,1);plot(f,abs(Xf),'r',f,Hf)

grid

axis([-2 2 0 5])%LIMITE LA REPRESENTATION DE X(f)ET H(f)

title('Representation fréqentielle de TF[x(t)]et de TF[h(t)]')

xlabel('Fréquence f')

ylabel('amplitude')

legend('X(f)=TF[x(t)]','H(f):filtre passe-bande idéal','de fréquence de coupure Fc=2/T')

%B) Représentation graphique de Y(f)=H(f)X(f)

subplot(2,1,2);plot(f,abs(Yf),'m')

grid

axis([-2 2 0 5]) % limite la représentation de Y(f) à l'intervalle de fréquences

[-2 2HZ]

title('Représentation fréquentielle :Y(f)')

xlabel('Fréquence f')

ylabel('amplitude')

legend ('Y(f)=H(f).X(f)')
```

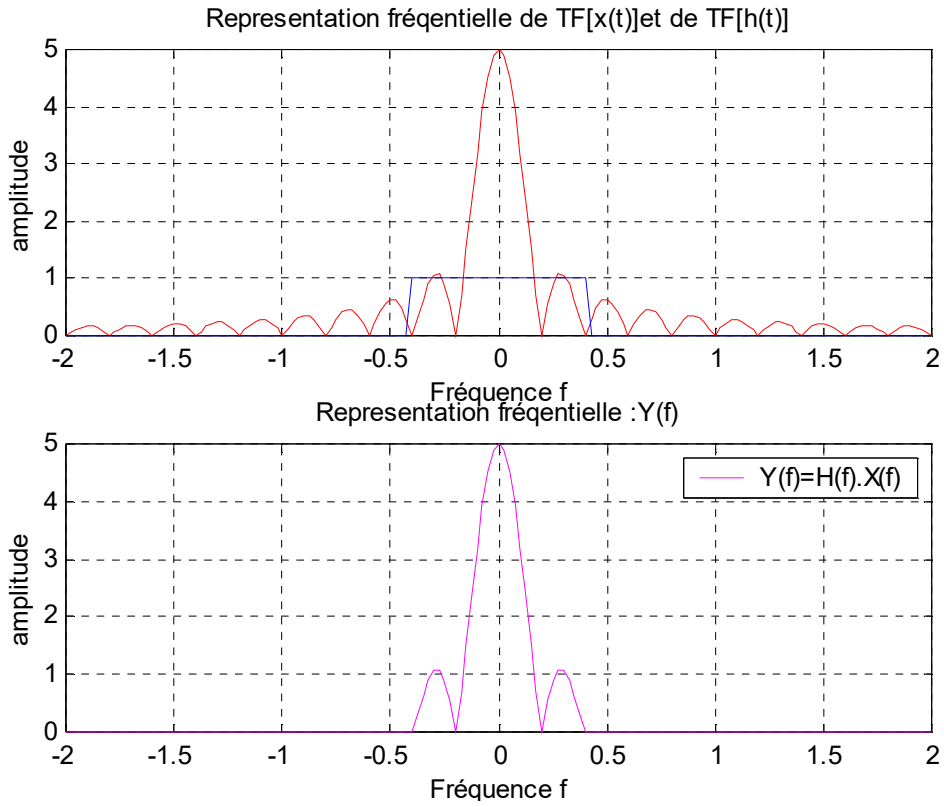


Figure 2.7 : représentation fréquentielle filtrée

Remarque :

On remarque que notre signal d'entrée X(f) est filtré par le filtre idéal H(f) de bande passante [-0.5 0.5], et la sortie Y(f) contient seulement un lobe principale et deux lobes secondaires dont les fréquences sont comprises entre -0.5 et 0.5.

Calcul et tracé de y(t)

Pour obtenir y(t), il nous faut revenir dans le domaine temporel. Pour cela, on utilise la Transformée de Fourier Inverse : $y(t)=TF^{-1}[Y(f)]$ .on utilise la commande ifft de Matlab.

```
%Calcul de y(t)=TF-1[Y(f)]

yt=abs(ifft(fftshift(Yf)/Te));

%Représentation graphique y(t)=h(t)*x(t)

figure(5)

plot(t,yt)

grid

axis([-20 20 0 1.2])

title('Représentation temporelle y(t)=TF[Y(f)]')
```

```
xlabel('temps t')
ylabel('amplitude ')
legend('y(t)=h(t)*x(t)')
```

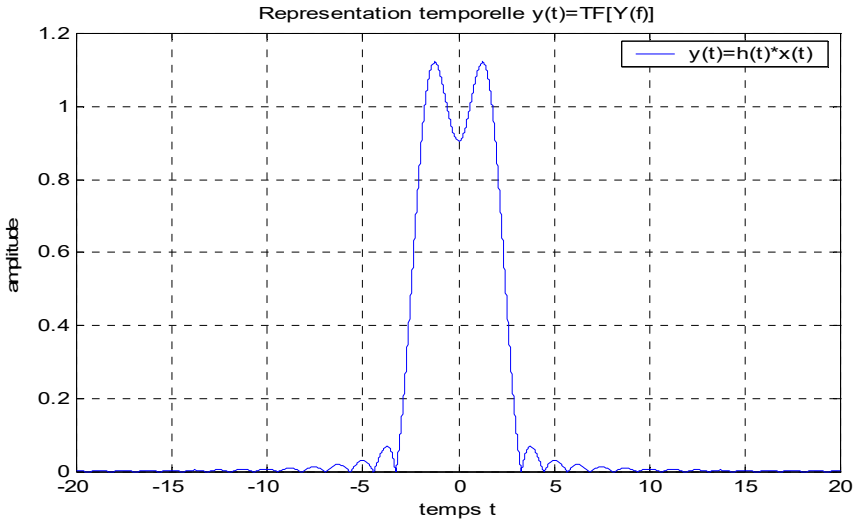


Figure 2.8: représentation temporelle y(t)

Remarque :

On observe que le signal à la sortie de filtre n'est pas le même que le signal à l'entrée ; la porte est déformée, à cause de filtrage des fréquences hors la bande [-0.5 0.5].

Tracé de y(t) pour des valeurs croissantes de la fréquence de coupure

```
%Definition des différentes valeurs de fc pour lesquelles on étudiera y(t)
fc=[2/T,1,5,15,50];
figure(6)
for k=1:5
   Hf=((-fc(k)<=f)&(f<=fc(k)));
   Yf=Hf.*Xf;
yt=abs(ifft(fftshift(Yf)/Te));
subplot(5,1,k);plot(t,yt);
grid
axis([-10 10 -0.05 1.5])
if (k==1)
```

```
title('Représentation du signal d''entrée x(t) après un filtrage passe-bande idéal h(t):y(t)
=h(t)*x(t)')

end

xlabel('temps t')

ylabel('amplitude ')

switch k

   case 1,legend('fc=2/T HZ')

   case 2,legend('fc=1 HZ')

   case 3,legend('fc=5 HZ')

   case 3,legend('fc=15 HZ')

   case 3,legend('fc=50 HZ')

end

end
```

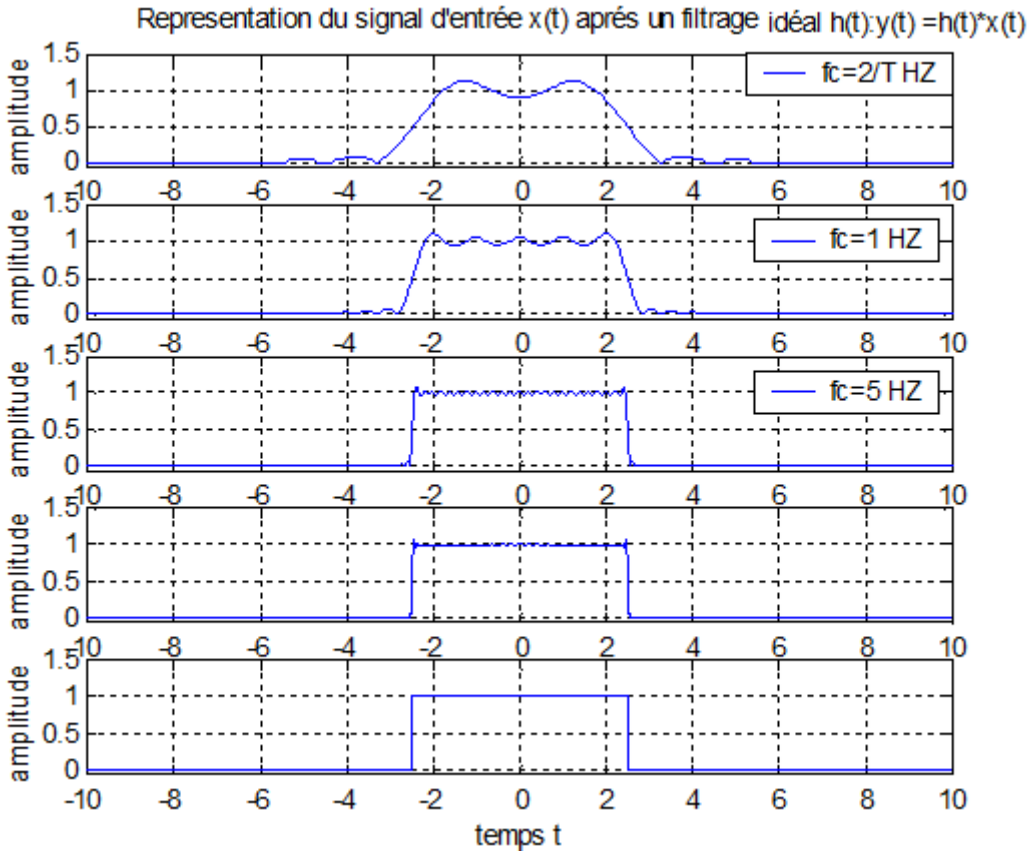


Figure 2.9 : représentation du signal d'entrée

Remarque :

On constate que lorsque la fréquence de coupure d'un filtre idéal augmente, le signal reconstitué à la sortie de filtre est de même forme que le signal avant le filtrage. Donc le choix de la fréquence de coupure d'un filtre est très important pour un filtrage idéal.

Travail demandé

Donner le programme qui permet de donner la fonction du filtre passe bas et filtre passe haut ?

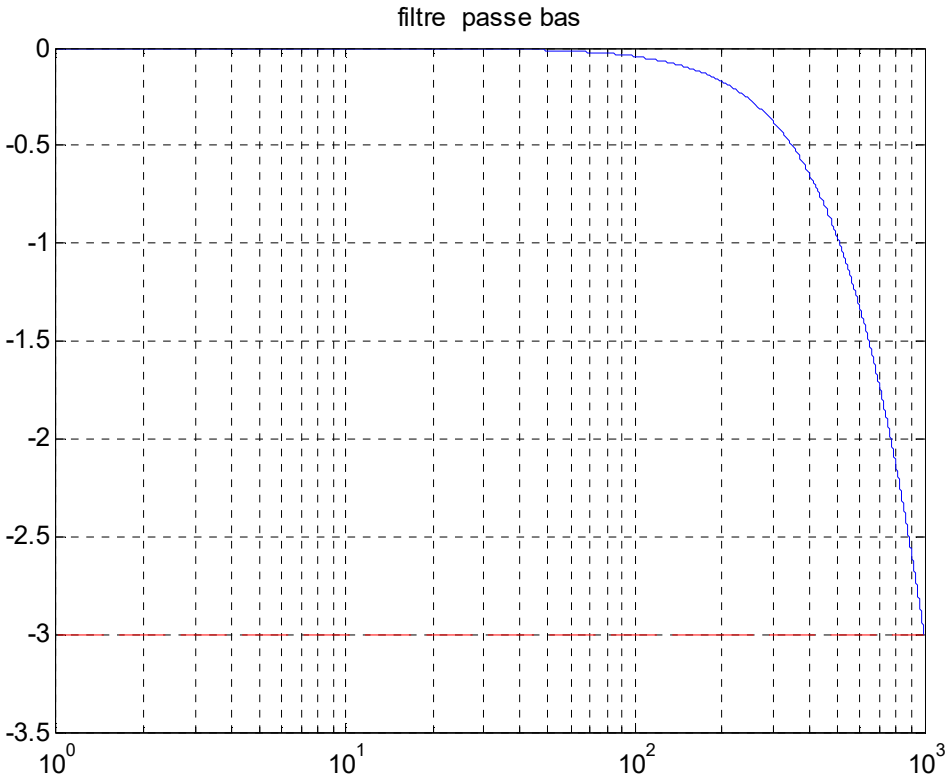
Solution

du Travail demandé

1-Notre objectif est de programmé la fonction du filtre passe bas sur **MATLAB**

Programme :

```
clear all
close all
clc
fe=1000 ;%fréquence d'échantillonnage
fc=62.5;%frquence de coupure
te=1/fe;%période d'échantillonnage
t=[0:te:1];
freq=(0:length(t)-1)*(fe/length(t));
w=2*pi*freq;
wc=2*pi*fe;
x=w/wc;
H=1./(1+i*x);
Module=20*log10(abs(H));
figure
semilogx(freq,Module,'')
hold on
semilogx(freq,max(Module)-3*ones(1,length(H)),'r--')
grid
title ('filter  passe bas')
```



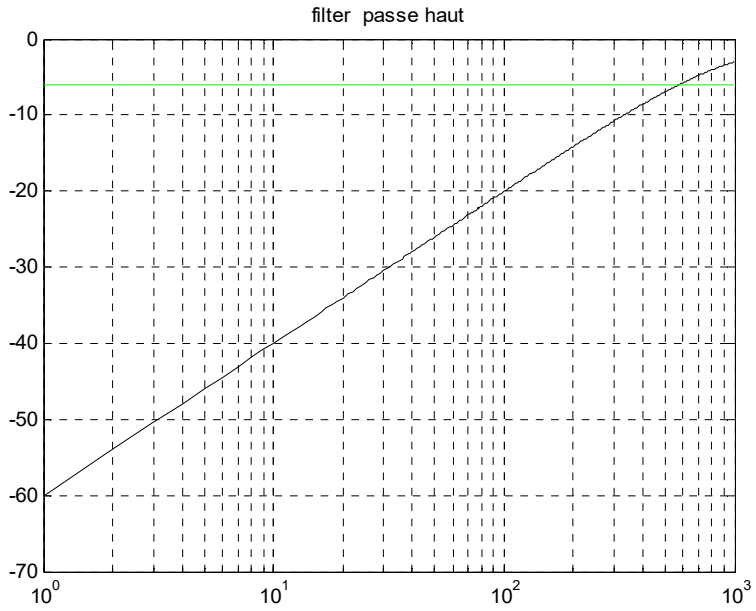
1-2- Filtre passe haut :

Notre objectif est de programmé la fonction du filtre passe haut sur **MATLAB**

Programme :

```
clear all
close all
clc
fe=1000 ;%fréquence d'échantillonnage
fc=62.5;%frquence de coupure
te=1/fe;%période d'échantillonnage
t=[0:te:1];
freq=(0:length(t)-1)*(fe/length(t));
w=2*pi*freq;
wc=2*pi*fe;
x=w/wc;
H=(j*x)./(1+j*x);
Module=20*log10(abs(H));
```

```
figure
semilogx(freq,Module,'k')
hold on
semilogx(freq,max(Module)-3*ones(1,length(H)),'g')
grid
title ('filter  passe haut')
```



TP3: Transformée de Fourrier Discrète

1. Objectif du TP

L'objectif de ce TP est d'aborder les notions de base de la transformer de Fourier discrète. Puis, effectuer l'analyse spectrale des signaux non périodiques a l'aide des outils disponibles sous Matlab.

2. Partie théorique

2.1. ECHANTILLONNAGES ET QUANTIFICATION :

La plupart des signaux (du moins ceux qui ont une origine naturelle) sont intrinsèquement analogiques. Il est donc logique que les machines permettant de créer ou de modifier ces signaux aient longtemps été elles-mêmes exclusivement analogiques. Le téléphone en est un bel exemple. Ce sont ces signaux analogiques qui nous intéressent dans ce cours. Cependant, depuis les années 70, on dispose de systèmes électroniques (CAN : convertisseur analogique-numérique, ou ADC : analog-digital converter) permettant d'*échantillonner* et de *quantifier* les signaux analogiques, les transformant ainsi en signaux numériques.

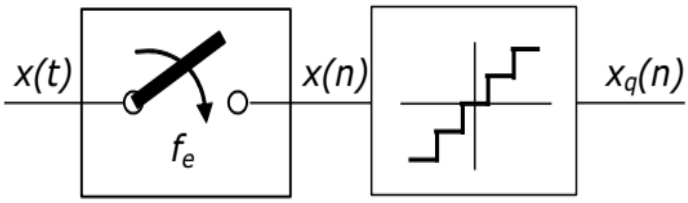
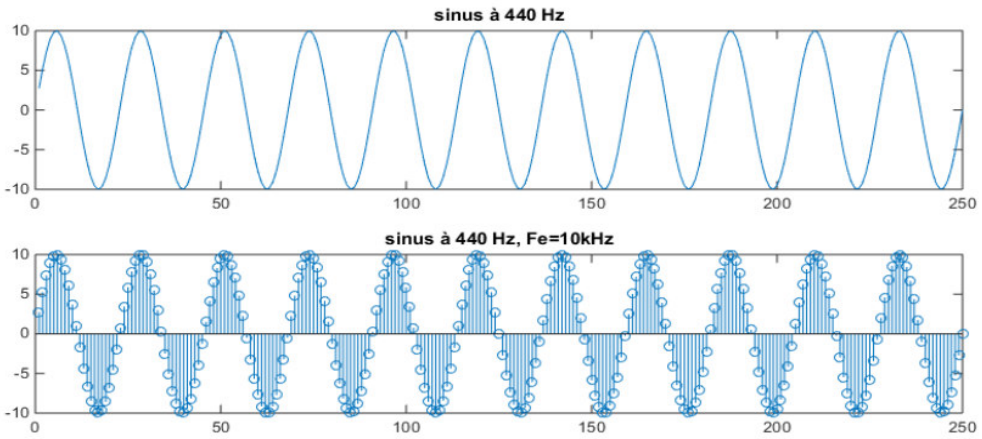


Figure 3.1. Représentation schématique de l'échantillonnage et de la quantification d'un signal analogique.

2.1. 1. Echantillonnage :

Exemple :

```
A=10; f=440; Fe=10000;
t=(1:20000)/Fe;
signal = A * sin(2*pi*f*t); % f=440 Hz, Fe=10000 Hz

soundsc(signal,10000);
subplot(2,1,1); plot(signal(1:250)); title('sinus à 440 Hz');
subplot(2,1,2); stem(signal(1:250)); title('sinus à 440 Hz, Fe=10kHz');
```

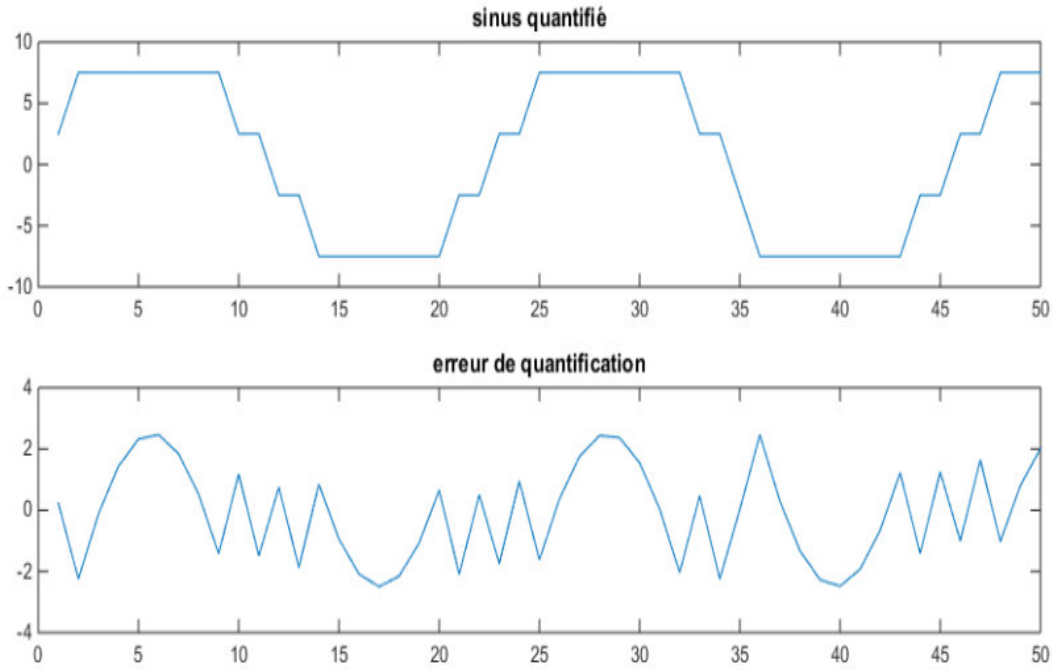


2.2. Quantification :

Exemple :

Quantification d'un signal sinusoidal

```
A = 10; f = 440; Fe = 10000;
signal = A * sin(2*pi*f*(1:20000)/Fe); % f=440 Hz, Fe=10000 Hz

b = 2;
quantizer_max = max(signal);
q = (2*quantizer_max) / 2^b;
signal_quantized = floor(signal/q)*q + q/2;
error = signal - signal_quantized;

soundsc(signal_quantized,10000);
plot(signal(1:50));
hold on;
subplot(2,1,1), plot(signal_quantized(1:50)); title('sinus quantifié');
subplot(2,1,2), plot(error(1:50)); title('erreur de quantification');
```



Il est à noter que, même sans cette opération formelle de quantification (ici à 2bits), les échantillons sont en réalité *déjà* stockés par MATLAB sous forme quantifiée.

3. Transformée de Fourier Discrète

Si la transformée de Fourier à temps discret est conceptuellement plus simple que la transformée de Fourier d'un signal analogique, il n'en reste pas moins que le calcul de la TFTD nécessite en principe une charge de calcul infinie, puisque la série qui la définit comporte un nombre infini de termes, et que l'estimation doit en être faite pour toutes les valeurs de la fréquence normalisée F entre 0 et 1.

C'est la raison pour laquelle la *transformée de Fourier Discrète* ou *TFD* (en anglais *Discrete Fourier Transform*, ou *DFT*) a été introduite. Son calcul est en effet limité à un nombre fini de valeurs de n et pour un nombre fini de valeurs de F.

3.1. Définition

Considérons une suite finie de N échantillons $\{x(n)\}=\{x(0),x(1),\ldots,x(N\text{-}1)\}$. On définit sa *transformée de Fourier Discrète* comme la suite $\{X(k)\}$ voir figure :

$$X(k)=\sum_{n=0}^{N-1}x(n)e^{-j\pi(k\frac{2\pi}{N})} \quad (k=0\ldots N-1)$$

Il ne s'agit donc ni plus ni moins de la TFTD d'un signal numérique dont on ne considère que les N premiers échantillons, calculée pour les N pulsations normalisées $\boldsymbol{\varphi} = \left\{\mathbf{0}, \frac{\mathbf{2\pi}}{\boldsymbol{N}}, \frac{\mathbf{4\pi}}{\boldsymbol{N}}, \dots\dots\dots, \frac{\mathbf{2(N-1)\pi}}{\boldsymbol{N}}\right\}$,

c-à-d pour les N fréquences normalisées $\boldsymbol{F} = \{\mathbf{0}, \frac{\mathbf{1}}{\boldsymbol{N}}, \frac{\mathbf{2}}{\boldsymbol{N}}, \dots\dots\dots, \frac{\mathbf{(N-1)}}{\boldsymbol{N}}\}$.

Nous verrons plus loin que la transformée de Fourier Discrète inverse est donnée par :

$$x(n) = \frac{1}{N}\sum_{k=0}^{N-1} X(k) e^{jnk\frac{2\pi}{N}} \quad (n = 0 \dots N-1)$$

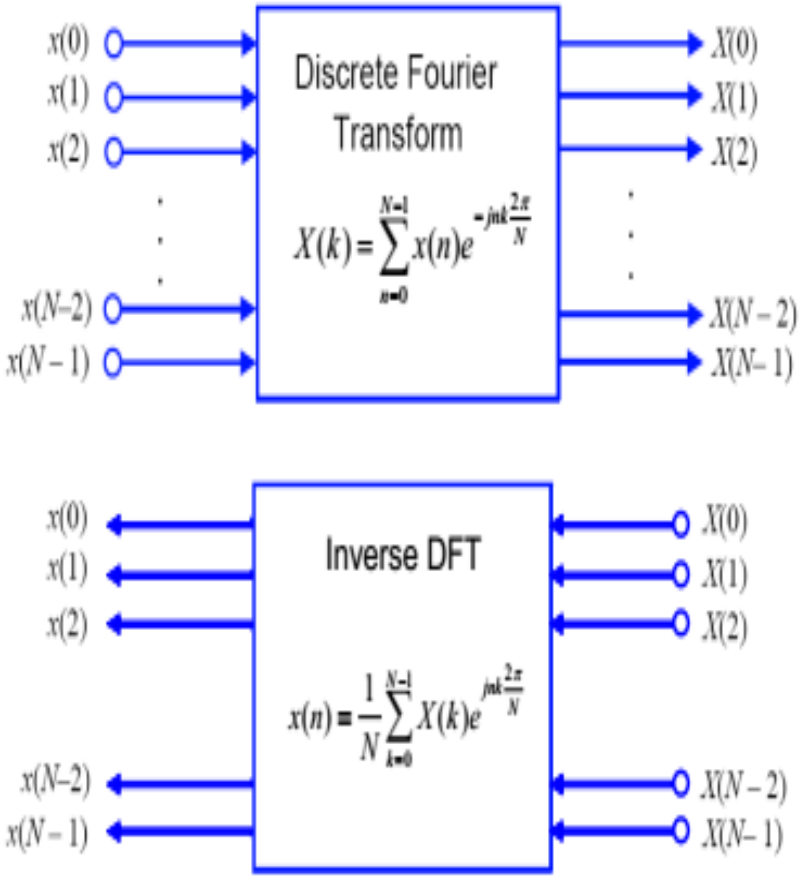


Figure 3.2. TFD et TFD inverse

3.2. Signaux numériques :

L'´échelon unité

Pour le cas de l'´échelon unité, on se contentera d'un nombre fini d'´échantillon. On peut ´écrire le programme suivant :

```
%echelon unite
t=-10:10;
x=[zeros(1,10),
ones(1,11)];
 stem(t,x);
 axis([-10 10 -0.5 1.5]);
title('Echelon unit´e');
xlabel('n');
ylabel('Amplitude');
```

Sinus et exponentielle décroissante

On recommence avec la fonction suivante $\sin(0,35 \times n)$. On écrira le programme suivant :

```
1-sinus
%sinus t=-10:10;
x=sin(0.35*t);
stem(t,x);
axis([-10 10 -1.5 1.5]);
title('sinus'); xlabel('n');
ylabel('Amplitude');
```

Exponentielle

Ensuite avec une exponentielle décroissante $e^{0,2\times n} \times u[n]$ avec u l'échelon unité. On a le programme suivant :

```
%exponentielle
 t=-10:10; u=[zeros(1,10),ones(1,11)]; x=exp(-0.2*t).*u;
stem(t,x);
axis([-10 10 -1.5 1.5]);
title('Exponentielle retarde');
```

```
xlabel('n');
ylabel('Amplitude');
```

Exemple de TFD :

Considérons la TFD de la partie imaginaire (le sinus) d'un des vecteurs de base, par exemple ***w2*** pour *N*=8. Puisque Im(***w2***)= -j/2 (***w2-w-2***)= -j/2 (***w2-wN-2***), toutes les valeurs *X(k)* de la TFD seront nulles, sauf *X(2)* et *X(6)* qui vaudront –4j et +4j, ce qui correspond bien à une suite conjuguée modulo *N*. On obtient effectivement ce résultat sous Matlab :

```
x=sin(2*pi/8*2*(0:7));
X=fft(x,8)
subplot(3,1,1); stem(x);
subplot(3,1,2); stem(abs(X));
subplot(3,1,3); stem(angle(X));
```

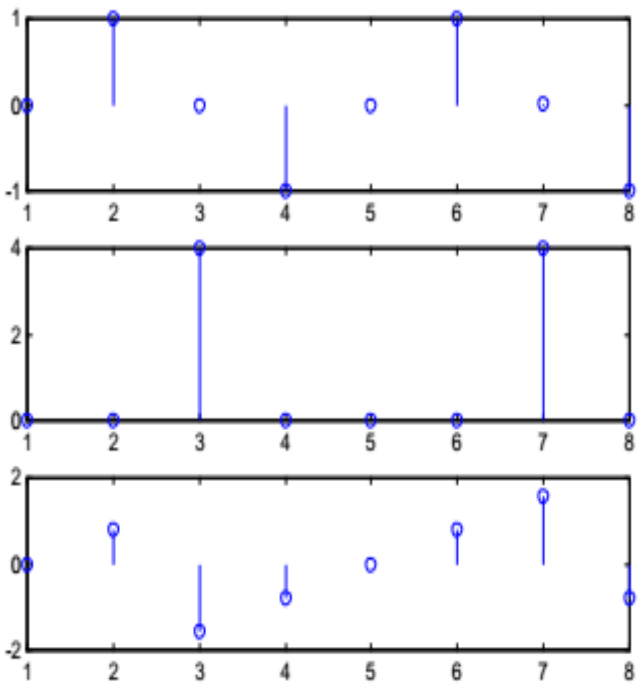


Figure 3.3. Module et argument de Im(***w2***) pour *N*=8 : suite conjuguée modulo 8

4. Transformée de Fourier Rapide

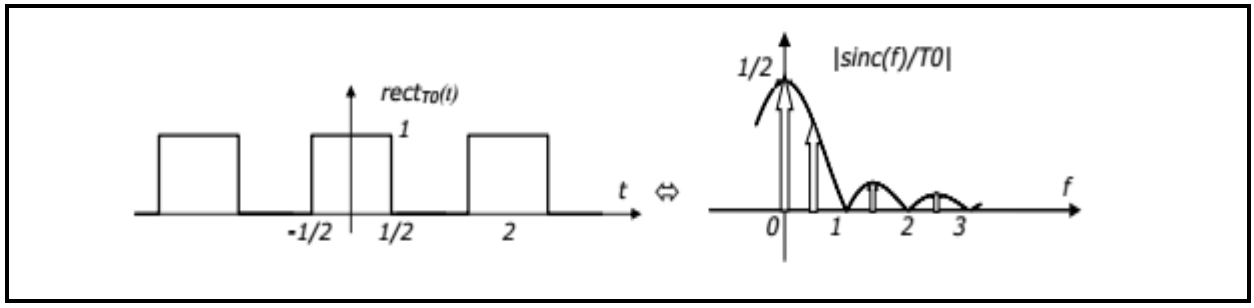
En 1965, Cooley et Tukey proposèrent une méthode qui permet de réduire considérablement le temps de calcul de la TFD d'une suite dont le nombre d'échantillons *N* est décomposable en

facteurs (typiquement, une puissance de 2). Par la suite, de nombreux algorithmes ont été publiés; ils sont connus sous le terme général de *transformation de Fourier rapide* (*TFR ;* ou *FFT : Fast Fourier Transform*). Tous ces algorithmes sont basés sur un même principe qui consiste à décomposer le calcul de la TFD en plusieurs TFD de longueur plus petite. La mise en œuvre de ce principe conduit à différentes méthodes dont les performances sont comparables. Nous décrirons ici plus particulièrement l'algorithme dit *radix* 2 avec entrelacement dans le temps.

4.1. Signaux périodique :

Exemple – signal rectangulaire périodique

Rendons le signal rectangulaire rect(t) périodique de période égale à 2s. sa transformée de Fourier fera apparaitre des raies en f=0, 1/2, 1, 3/2,ect, de valeur sinc(f)/2



On constate que la transformée fait apparaitre une raie en f=0 (composante continue=1/2), puis uniquement des harmoniques *F** impaires, d'amplitudes décroissant en 1/k.

```
signal=[rectwin(10) ; zeros(10,1)]; % rectangle width : 1 s = 10 samples
signal = [signal; signal; signal]
[fourier,frequencies]=freqz(signal) % computes the FT in f=[0,Fe/2]

subplot(211); stem(signal); title('f(n)')
Fe=10;
subplot(212); plot(frequencies/pi*(Fe/2),abs(fourier)); title('|F(f)|');
```

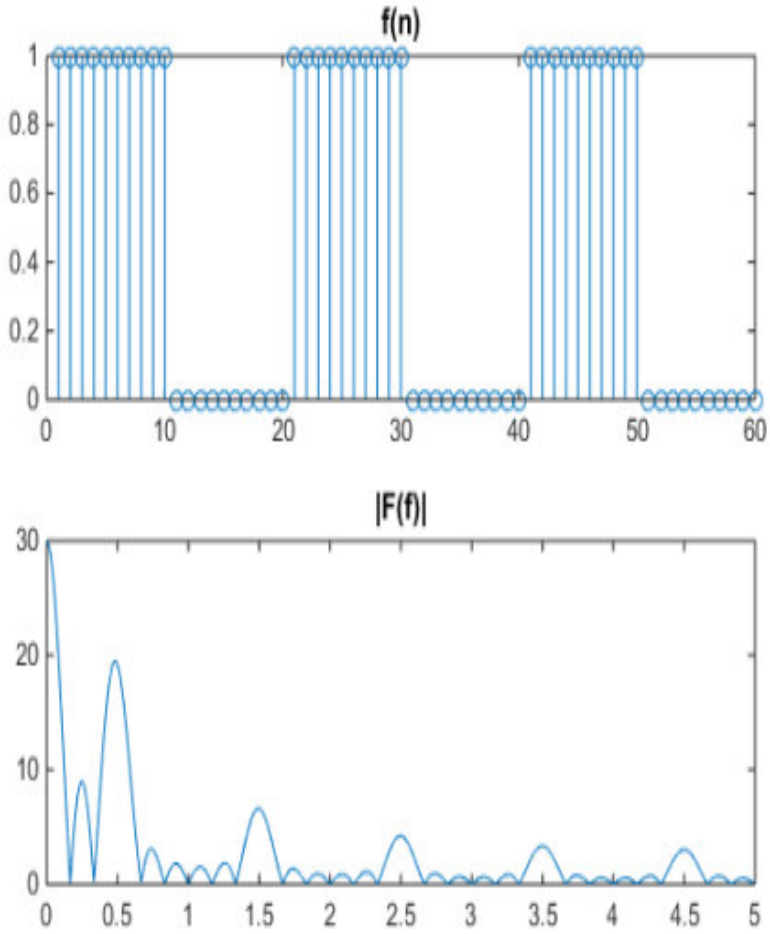


Figure 3.4.TFD signal rectangulaire périodique

On retrouve bien ici les raies attendues, avec une amplitude multipliée par le N d'échantillons retenus (ici60).

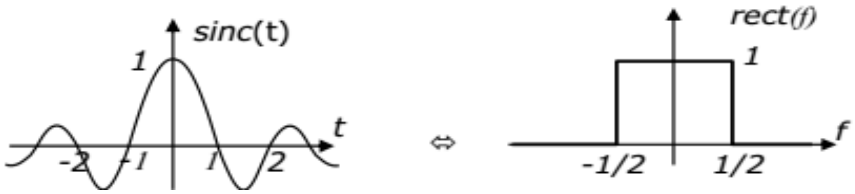
4.2. Transformée de Fourier discrète signal non périodique

Si le signal dont on cherche à observer le spectre est de durée finie, et que le nombre *N* d'échantillons dont on dispose couvre une plage temporelle supérieure ou égale à cette durée, l'effet de N est évidemment transparent : la TFD affiche alors *N* valeurs de la TFTD originale.

Exemple 1 – Transformée du sinus cardinal

Par application de la propriété de dualité de la transformée de Fourier, il vient :

$$\mathrm{sinc}(t) \overset{\mathcal{F}}{\Leftrightarrow} = \mathrm{rect}(-f) = rect(f)$$




```
Fe=10 ; t=(-100:99)* 1/Fe;    %Fe = 10 ; 200 samples ; 20 oscillations
signal=sinc(t),
[fourier,frequencies]=freqz(signal) % computes the FT in f=[0,Fe/2]

subplot(211); stem(signal); title('f(n)')
Fe=10;
subplot(212); plot(frequencies/pi*(Fe/2),abs(fourier)); title('|F(f)|');
```

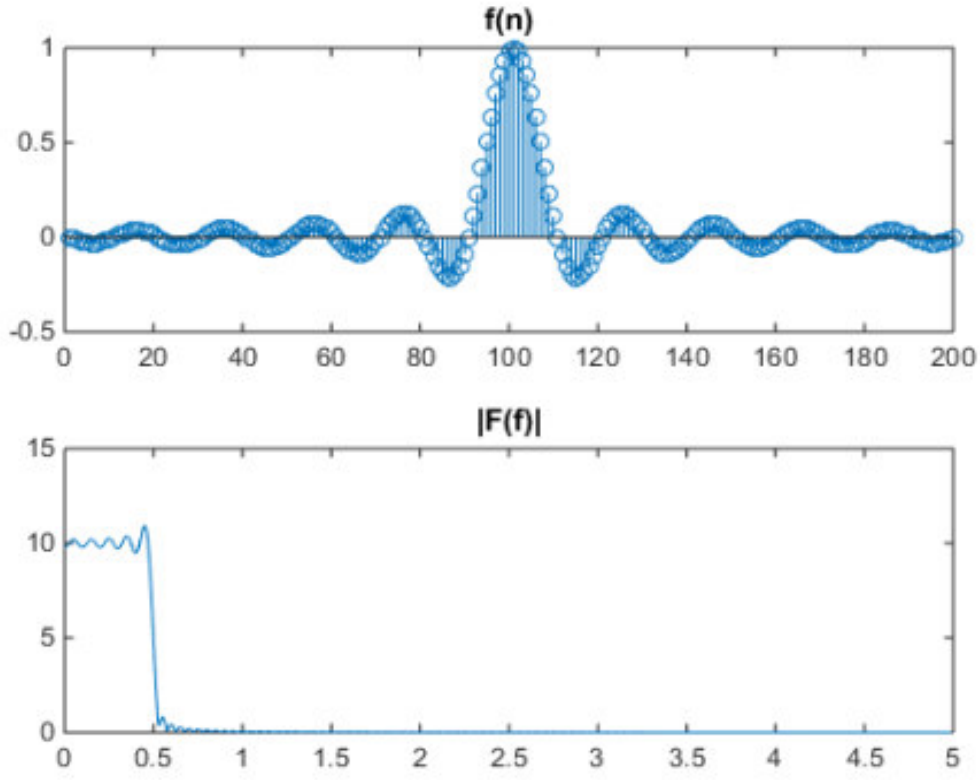


Figure 3.5. Transformée du sinus cardinal

Travail demandé

1- L'échantillonnage :

On échantillonne une sinusoïde f(t)=sin ($2\pi f_0 t$) a la fréquence d'échantillonnage 10000hz. Dessinons l'allure des échantillons (c'est-à-dire l'allure de la fonction f*(t)correspondante) pour des valeurs de f_0 égales à : 1000hz, 5000hz, 7500hz, 9000hzet 31000hz.

Pour que les graphiques possèdent des axes temporels identiques, choisissons de montrer les 10 premières ms de signaux.

Donner le programme ?

2-TFD

Considérons la TFD de la partie imaginaire (le sinus) d'un des vecteurs de base, par exemple ***w2*** pour *N*=8. Puisque Im(***w2***)= -j/2 (***w2-w-2***)= -j/2 (***w2-wN-2***), toutes les valeurs *X(k)* de la TFD seront nulles, sauf *X(2)* et *X(6)* qui vaudront –4j et +4j, ce qui correspond bien à une suite conjuguée modulo *N*.

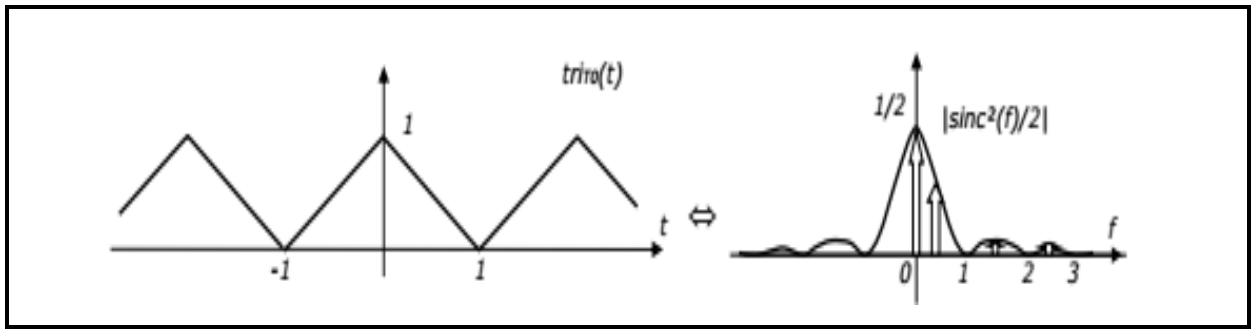
Donner sous Matlab TFD de sin ?

3-TF Signaux périodique *:*

Signal triangulaire périodique

Rendons le signal rectangulaire tri(t) périodique de période égale à 2s. Sa transformée de Fourier apparaitre des raies en f=0,1/2, 1, 3/2,etc, de valeur $sinc^2(f)/2$

Donner le programme qui représente sa transformée de Fourier de ce signal ?



4-Transformée de Fourier discrète signal non périodique

Calculons la TFTD d'une fenêtre rectangulaire de 16 points, par FFT sur 16, 8, et 4 points :

Solution
du Travail demandé

1-Soit le programme suivant :

```
subplot(6,1,1); stem(sin(2*pi*1000*[0:99]/10000));
subplot(6,1,2); stem(sin(2*pi*2500*[0:99]/10000));
subplot(6,1,3); stem(sin(2*pi*5000*[0:99]/10000));
subplot(6,1,4); stem(sin(2*pi*7500*[0:99]/10000));
subplot(6,1,5); stem(sin(2*pi*9000*[0:99]/10000));
subplot(6,1,6); stem(sin(2*pi*31000*[0:99]/10000));
```

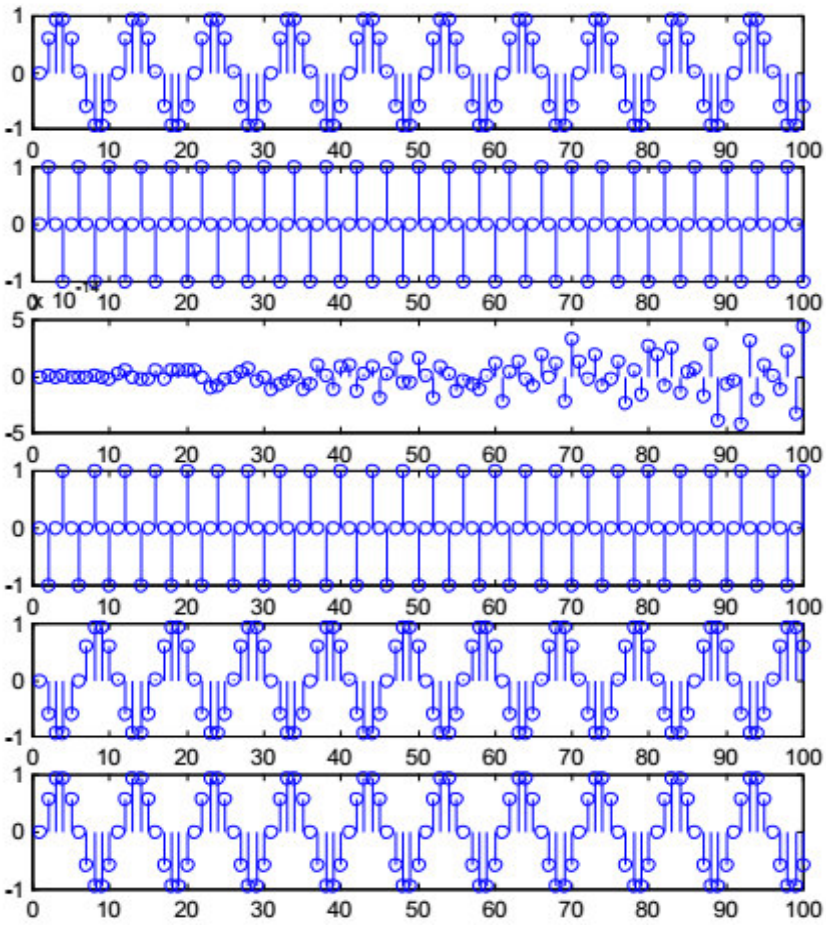


Figure 3.6. Echantillonnage de sinusoïdes

On constate :

- Que seuls les échantillonnages des sinusoïdes à 1000 Hz et 2500 Hz donnent une image réaliste des signaux sous-jacents (on peut retrouver la fréquence des sinusoïdes correspondantes en mesurant leur période sue le graphique).
- Que l'échantionnage de la sinusoïde à 5000 Hz fait apparaitre un signal d'amplitude très petite. En théorie, les échantillons devraient être tous nuls. Les valeurs non nulles apparaissant sur le graphique sont le résultat d'erreurs d'arrondis de Matlab.
- Que les échantionnages des sinusoïdes à 7500 Hz et 9000Hz donnent des résultats identiques (au signe près) à ceux des sinusoïdes à 2500 Hz et 1000Hz. Il est par conséquent impossible, après échantillonnage, de retrouver la fréquence exacte des sinusoïdes sous-jacentes. C'est le résultat du repliement spectral des sinusoïdes de départ autour de la fréquence de Nyquist (5000Hz).
- Que l'échantillonnage de la sinusoïde à 31000 Hz donne exactement le même résultat que celui d'une sinusoïde à 1000Hz. C'est le résultat de la superposition de toutes les translatées (à des multiples de fe) de la transformée de Fourier du signal d'origine.

Un moyen simple d'éviter le repliement spectral est d'échantillonner à une fréquence supérieure à deux fois la plus haute composante fréquentielle du signal (théorie de Shannon). Certains signaux cependant (typiquement, les fonctions qui présentent des discontinuités), ont un spectre théoriquement infini. Dans ce cas, on peut toujours choisir la fréquence d'échantillonnage de façon à imposer que le recouvrement spectral soit inférieur à un seuil.

2-TFD

On obtient effectivement ce résultat sous Matlab :

```
x=sin(2*pi/8*2*(0:7));
X=fft(x,8)
subplot(3,1,1); stem(x);
subplot(3,1,2); stem(abs(X));
subplot(3,1,3); stem(angle(X));
```

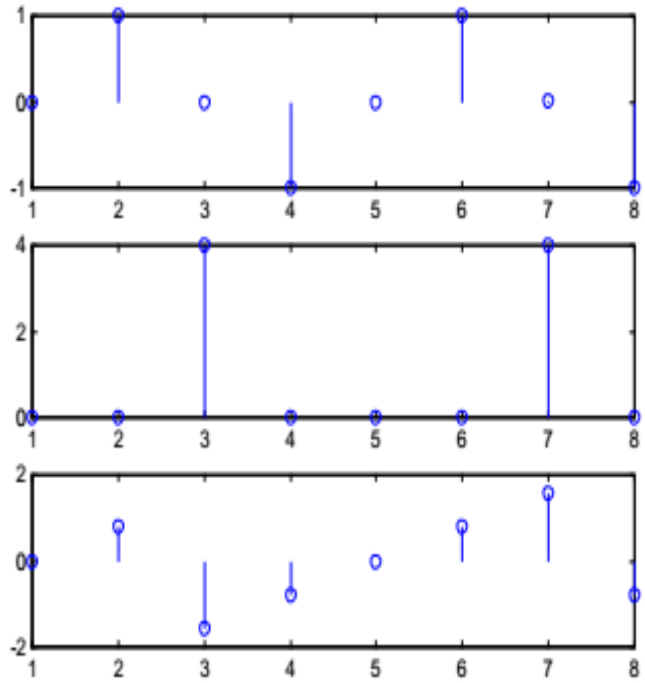
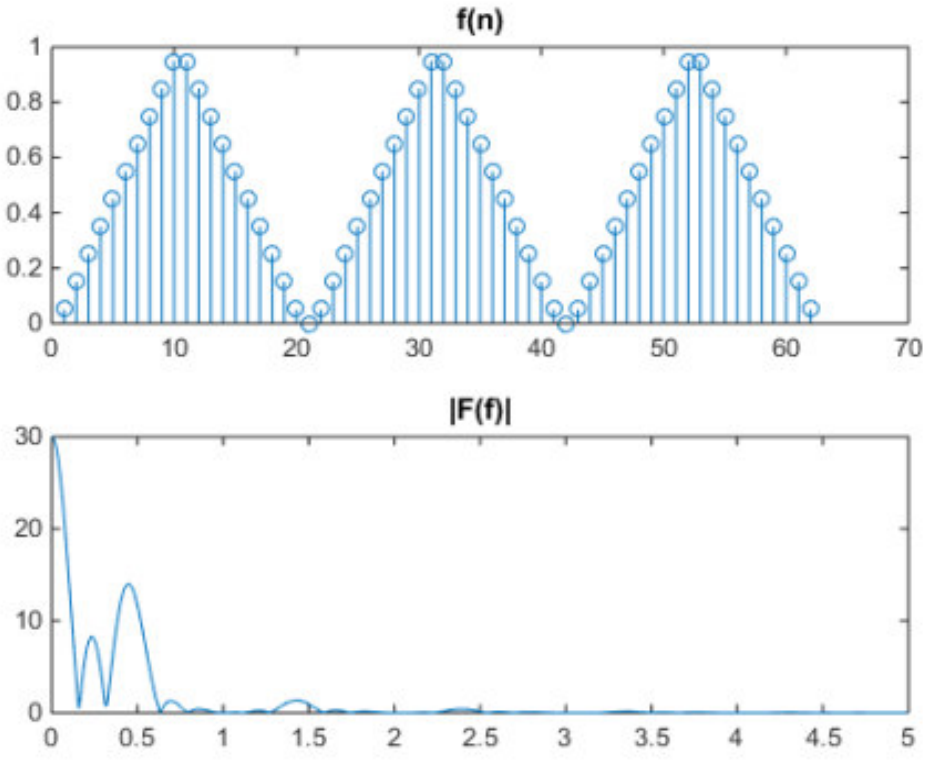


Figure3.7. Module et argument de Im(***w2***) pour *N*=8 : suite conjuguée modulo 8

3-

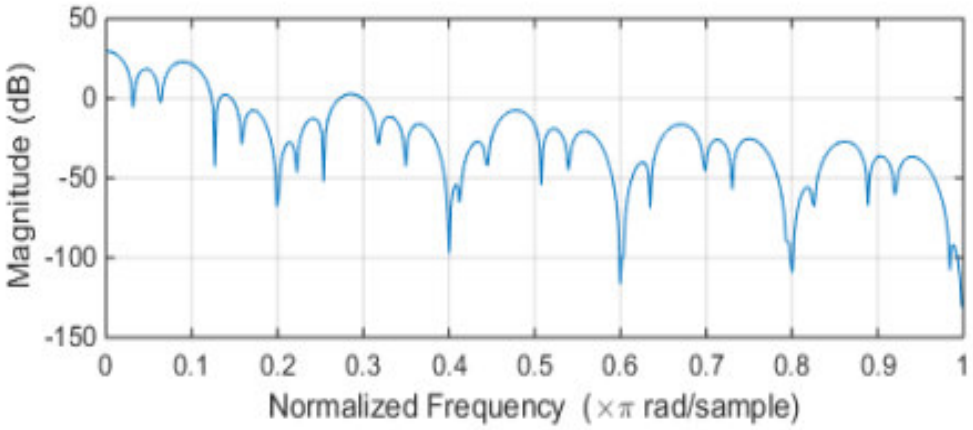
```
signal=[triang(20)]; % triangle width : 2 s = 20 samples
signal = [signal; signal; signal]
[fourier,frequencies]=freqz(signal) % computes the FT in f=[0,Fe/2]

subplot(211); stem(signal); title('f(n)')
Fe=10;
subplot(212); plot(frequencies/pi*(Fe/2),abs(fourier)); title('|F(f)|');
```

On constate que la transformé fait apparaitre une raie en f=0 (composante continue=1/2), puis uniquement des harmoniques f*impaires, d'amplitudes décroissant en 1/k.



Comme les amplitudes des raies sont petites, on peut avantageusement afficher le module de la transformée de Fourier en dB. c'est ce que fait par défaut la fonction freqz, sur un axe de fréquence normalisé entre 0 et 1.

```
freqz(signal)
```



4-

```
x=ones(1,16);
subplot(6,1,1); stem(x);
subplot(6,1,2); stem((0:15)/16, abs(fft(x,16)));
x=[ones(1,8)+ones(1,8)];
subplot(6,1,3); stem(x);
subplot(6,1,4); stem((0:7)/8, abs(fft(x,8)));
x=[ones(1,4)+ ones(1,4)+ ones(1,4)+ ones(1,4)];
subplot(6,1,5); stem(x);
subplot(6,1,6); stem((0:3)/4, abs(fft(x,4)));
```

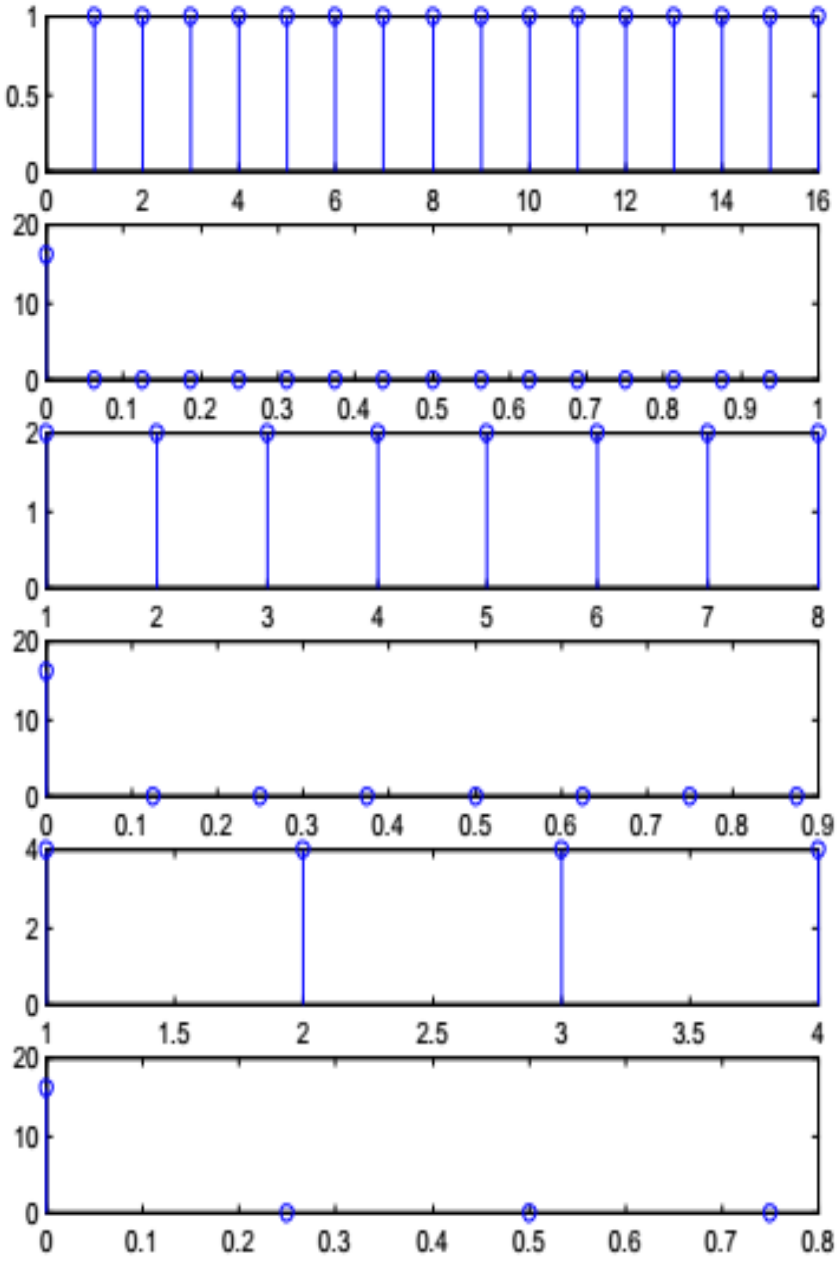


Figure3.8. FFT sur 16, 8, et 4 points de 16 échantillons d'une fenêtre rectangulaire.

TP4- Filtrage Numérique RII

Objectif du TP :

Ce TP s'appuie sur la notion de processus linéaire à réponse impulsionnelle infini(RII), on alterne des programmes de simulation de la réponse impulsionnelle par deux méthodes : par la commande : « **filter** » et la commande « **impz** ». , aussi la réponse du système selon une entrée bruitée et la représentation des pôles et des zéros.

1. Parties théorique :

1.1.Fonction de transfert :

Une fonction de transfert est une description entrée- sortie d'un système linéaire invariant .Dans ce TP en prend comme processus linéaire : un filtre numérique dont la fonction de transfert est définit dans le domaine fréquentiel (Transformée en Z).

1.2.Définition d'un filtre numérique :

Un filtre numérique peut être défini par une équation aux différences, c'est-à-dire l'opération mathématique du filtre dans le domaine temporel (discret).La forme générale du filtre d'ordre M est la suivante :

$$y[n] = \sum_{k=0}^{N} b_k \, .x[n-k] - \sum_{k=1}^{M} a_k \, .y[n-k]$$

$$y[n]=b_0.x[n]+b_1.x[n-1]+ b_2.x[n-2]+\ldots.+ b_N.x[n-N]- a_1.y[n-1]- a_2.y[n-2]+\ldots+ a_M.y[n-M]$$

La fonction de transfert générale d'ordre N d'un filtre numérique est la suivante :

$$H(z)=\frac{Y(z)}{X(z)}=\frac{\sum_{k=0}^{N} b_k .z^{-k}}{\sum_{k=0}^{M} a_k .z^{-k}}$$

$$H(z)=\frac{b_o+b_1.z^{-1}+b_2.z^{-2}+\cdots+b_N.z^{-N}}{a_o+a_1.z^{-1}+a_2.z^{-2}+\cdots+a_M.z^{-M}}$$

La valeur des coefficients a et b fixera le type de filtre, passe-bas, passe-haut, etc.

1.3. Classification des filtres numériques :

Il y a deux grandes familles de filtres numériques : la première, les filtres RIF, de l'anglais "Finite Impulse Response" (Filtre à réponse impulsionnelle finie). Les filtres RIF sont les plus populaires vu leur stabilité.

1.3.1. Filtre à réponse impulsionnelle finie (R IF)

Ces filtres peuvent être implantés en utilisant la formule suivante :

$$y(n) = \sum_{k=0}^{l} b(n)x(n-k)$$

Ce type de filtre est dit fini, car sa réponse impulsionnelle se stabilisera ultimement à zéro. Un filtre FIR est non récursif, c'est-à-dire que la sortie dépend uniquement de l'entrée du signal, il n'y a pas de contre-réaction. Ainsi, les coefficients a de la forme générale des filtres numériques sont tous égaux à zéro. Une propriété importante des filtres FIR est que les coefficients du filtre b sont égaux à la réponse impulsionnelle h du filtre. D'autre part, la forme temporelle du filtre est tout simplement la convolution du signal d'entrée x avec les coefficients (ou réponse impulsionnelle) b (ou h).

- **caractéristique des filtres à réponse impulsionnelle finie :**

  - Le filtre RIF à réponse impulsionnelle finie correspond à la structure précédente : l'échantillon de sortie y(n) dépend uniquement des échantillons entrées x(n), x(n-1), x(n-2), ...x(n-p).
  - Ce type de filtre est, par construction, toujours stable.
  - causals si leur réponse impulsionnelle g(k) est nulle pour k < 0. S'ils ne le sont pas, il suffit de décaler cette réponse.
  - Peuvent être à phase linéaire.

1.3.2. Filtre à réponse impulsionnelle infinie (IIR) :

En opposition, les filtres de la seconde famille, les IIR, de l'anglais "Infinite Impulse Response" (Filtre à réponse impulsionnelle infinie) possèdent une réponse impulsionnelle qui ne se stabilisera jamais, et ce, même à l'infini. Les filtres IIF peuvent être implantés en utilisant la formule suivante :

$$y[n] = \sum_{k=0}^{N} b_k \, .x[n-k] - \sum_{k=1}^{M} a_k \, .y[n-k]$$

Ce type de filtre est récursif, c'est-à-dire que la sortie du filtre dépend à la fois du signal d'entrée et du signal de sortie, il possède ainsi une boucle de contre-réaction ("feedback"). Les filtres IIR sont principalement la version numérique des filtres analogiques traditionnels : **Butterworth, Tchebychev, Bessel**, Elliptique.

- **caractéristique des filtres à réponse impulsionnelle infinie :**

- Plus flexibles que les filtres RIF : Le filtre RII à réponse impulsionnelle infinie correspond à une structure bouclée où l'échantillon y(n) dépend non seulement des x(n) mais aussi des y(n-p) : il y a donc un réinjection de la sortie vers l'entrée. Qui dit "rebouclage" dit "risque d'instabilité" (théorie de la contre réaction) avec une réponse impulsionnelle oscillatoire possiblement longue à se stabiliser d'où le nom RII.
- leur fonction de transfert est le rapport de deux polynômes en z.
- Nécessitent moins de calcul que les filtres RIF.
- Leur stabilité n'est pas garantie.
- position des pôles à vérifier.

1.4.La transformée en Z du produit de convolution:

Le produit de convolution est lié à la notion de filtrage sous deux conditions, à savoir la linéarité et l'indépendance du filtre vis à vis du temps (système invariant). A partir de ces deux conditions, l'opérateur de convolution peut être construit. Si l'on a un signal entrant e(t) et un élément filtrant ayant une fonction de transfert h(t) alors la transformée en Z du produit de convolution est donnée comme suit :

$$Y(t)=h(t)*e(t) \quad \xrightarrow{\text{T en Z}} \quad Y(z)=H(z).E(z)$$

Où ***Y***, ***E*** et ***H*** sont les transformées en ***Z*** des fonctions du temps ***y(t), e(t)*** et ***h(t)***.

2. Partie pratique :

Travail demandé +solution

1.2. Réponse impulsionnelle d'un filtre RII :

Exercice :

Soit un processus linéaire à réponse impulsionnelle infinie (filtre RII), décrit par la fonction de transfert numérique suivante :

$$H(z)=\frac{0.2+0.5z^{-1}}{1-0.2z^{-1}+0.8z^{-2}}$$

En utilisant la même approche que précédemment pour réaliser le script suivant :

PROG2 : réponse impulsionnelle d'un filtre R II et la représentation des pôles et des zéros

```
close all,clear all;
N=1024;
% nombes de points
imp=[1 0 0 0];
%l'entrée impulsion de Dirac
a=[0.2 0.5]
%initiation de vecteur des coéficients du numérateur
b=[1 0.2 0.8 8]
%initiation de vecteur des coéficients du déumérateur
x=randn(1,N);
%génération du bruit blanc
figure(1);
y=filter(b,a,x);
%générer la sortie du système ayant comme entrée un bruit
blanc
```

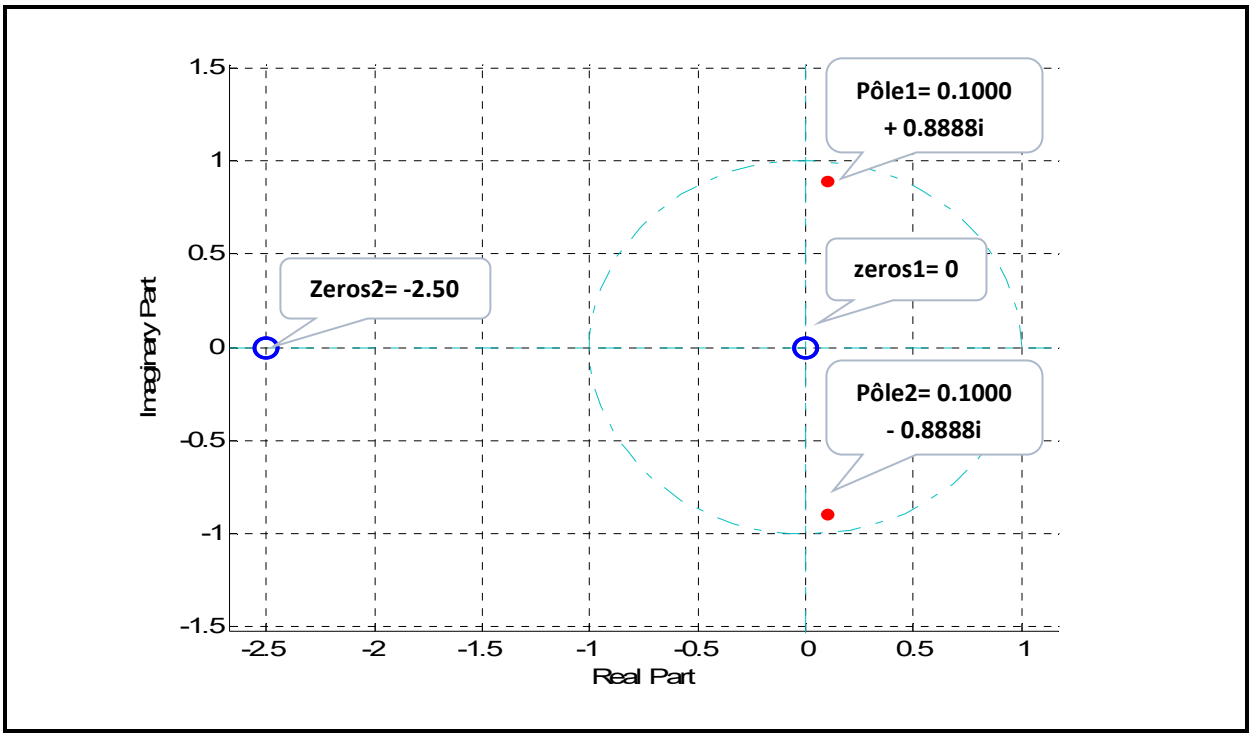


Figure 4.1-Représentation des pôles dans le plan Z

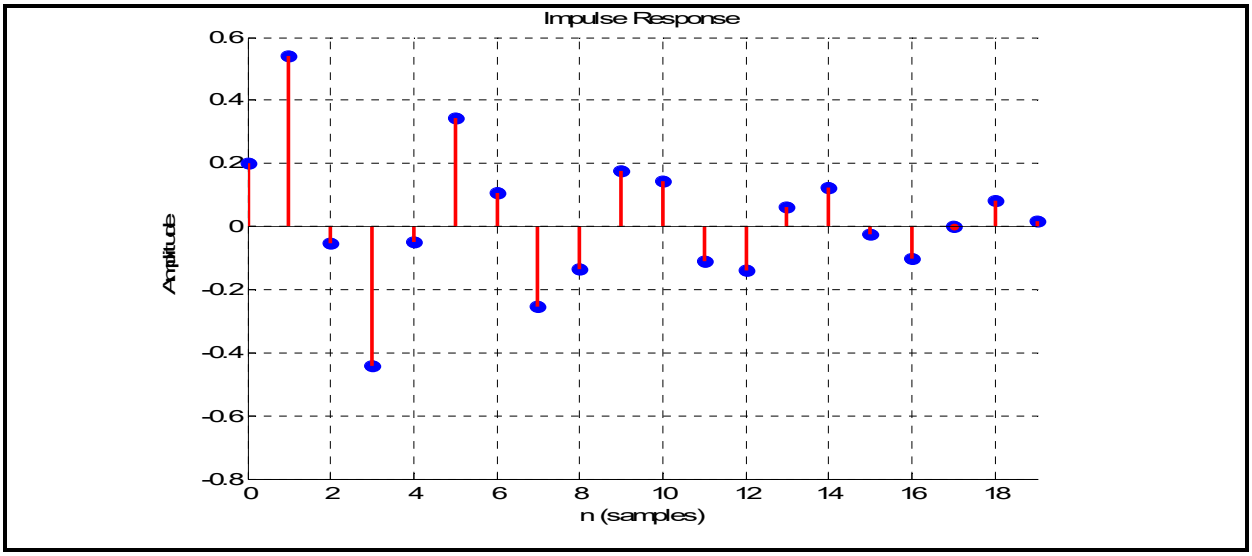


Figure 4.2-Représentation de la réponse impulsionnelle du filtre RII sur les 20 premiers en utilisant la commande « impz »

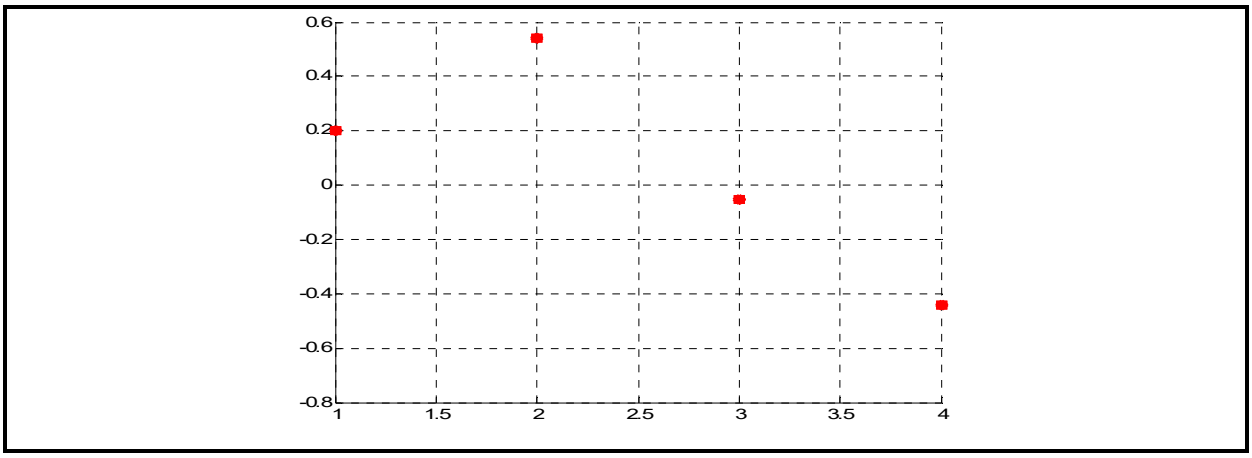


Figure 4.3- représentation de la réponse impulsionnelle du filtre RII sur les 20 premiers en utilisant la commande «filter »

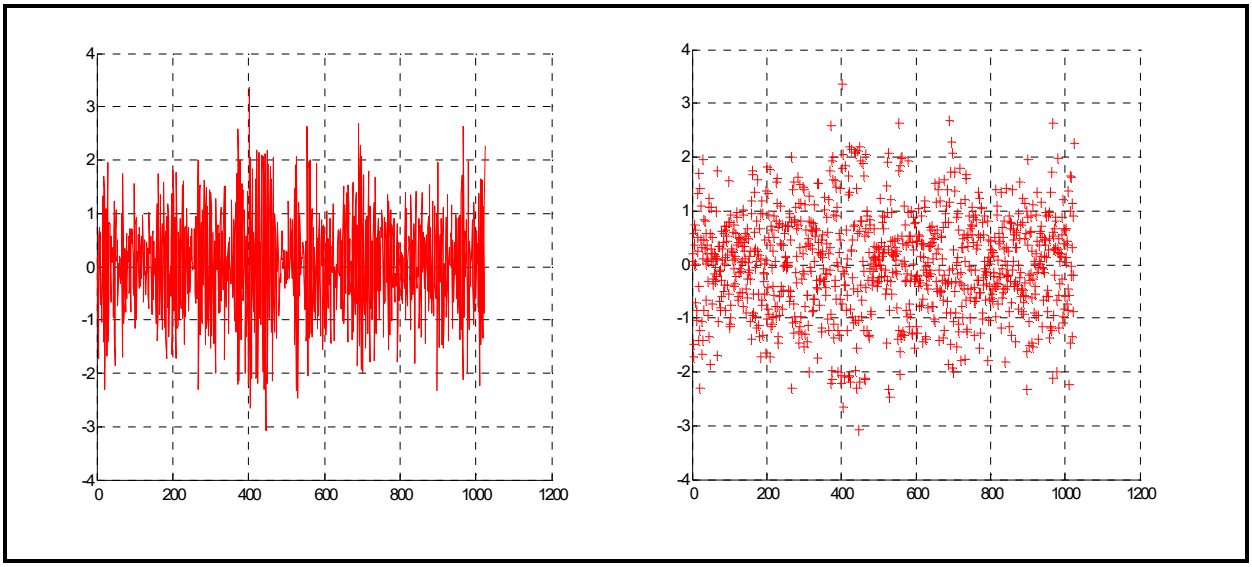


Figure 4.4- représentation de la sortie du système ayant comme entrée un bruit blanc

TP5- Filtrage Numérique RIF

1. Objectif du TP :

Ce TP s'appuie sur la notion de processus linéaire à réponse impulsionnelle finie (RIF), on alterne des programmes de simulation de la réponse impulsionnelle par deux méthodes : par la commande : « **filter** » et la commande « **impz** ». , aussi la réponse du système selon une entrée bruitée et la représentation des pôles et des zéros.

Travail demandé +solution

2. Filtre à Réponse impulsionnelle d'un filtre RIF

Un processus linéaire à réponse impulsionnelle finit (filtre FIR), décrit par la fonction de transfert suivante :

$$H(z)=0.5+0.75z^{-1}+1.2z^{-2}$$

Pour ces filtres, on étudiera la réponse impulsionnelle, le module et la phase de la fonction de transfert en fréquences, ainsi que la représentation des pôles et des zéros. la réponse impulsionnelle peut être obtenue en filtrant une impulsion par la commande **« filter »** ou bien directement à l'aide de la commande **« impz ».**on calculera seulement quelques dizaines de points de la réponse impulsionnelle des filtres FIR. On réalise le script suivant qui nous affichera les 4 graphes présentés dans les figures qui suivent (fig 4.1) :

PROG1 : réponse impulsionnelle d'un filtre FIR et la représentation des pôles et des zéros

```
close all,clear all;
N=1024;
% nombres de points
imp=[1 0 0 0];
%l'entrée impulsion de Dirac
a=1;
%initiation de vecteur des coéficients du numérateur
```

```
b=[0.5 0.75 1.2 0];
%initiation de vecteur des coéficients du déumérateur
x=randn(1,N);
%génération du bruit blanc
y=filter(b,a,x);
%générer la sortie du système ayant comme entrée un bruit blanc
resptf=filter(b,1,imp);
%générer la réponse impulsionnelle
figure(1);zplane(b,a);
%représentation des pôles et des zéros dans le plan Z
figure(2);
impz(b,1,20);
%calcule de la réponse impulsionnelle, avec 20 est la nombre de points demandés par la
réponse impulsionnelle
figure(3);
plot(resptf,'*');
%représentation de la sortie du système ayant comme entrée une impulsion
figure(4);plot(y);
%représentation de la sortie du système ayant comme entrée un bruit blanc
```

Le calcul des valeurs des zéros leurs modules et leurs phase sous Matlab donne :

```
>> zeros
zeros =
  -0.7500 + 1.3555i
  -0.7500 - 1.3555i
>> r=abs(zeros);% module d'un nombre complexe
r =
    1.5492
    1.5492
>> theta=angle(zeros), % argument d'un nombre complexe
theta =
    2.0762
   -2.0762
```

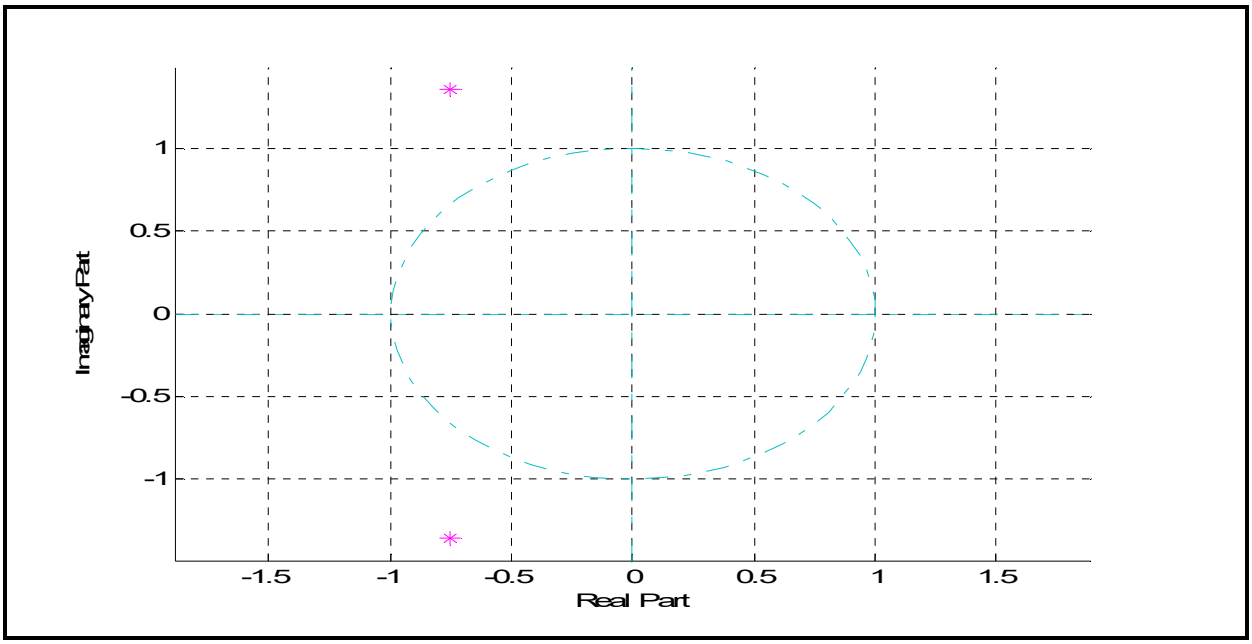


Figure 5.1-Représentation des zeros dans le plan Z

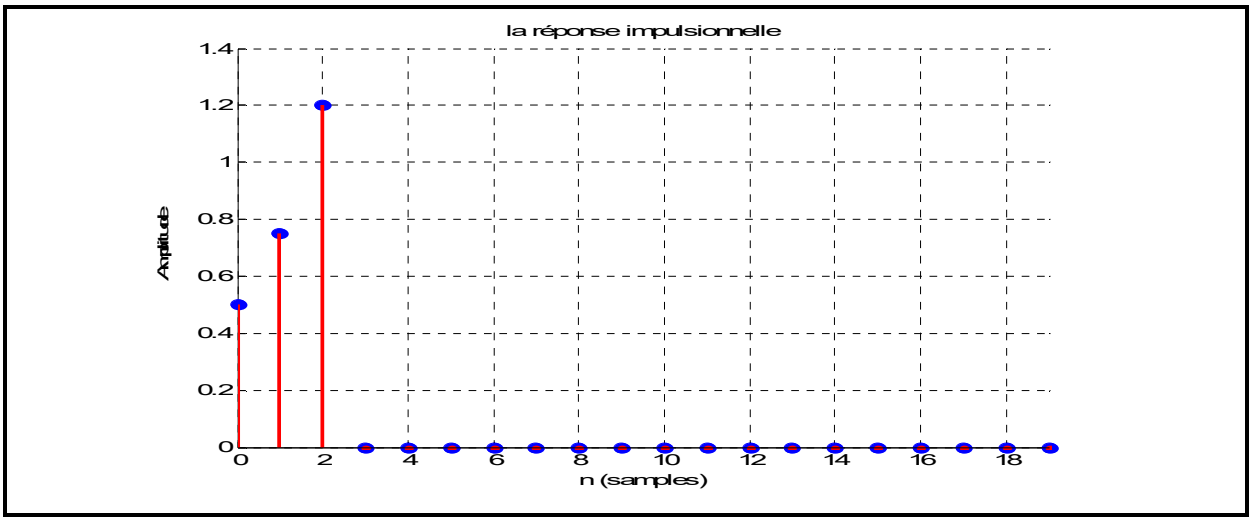


Figure 5.2-Représentation de la réponse impulsionnelle du filtre RIF sur les 20 premiers en utilisant la commande « impz »

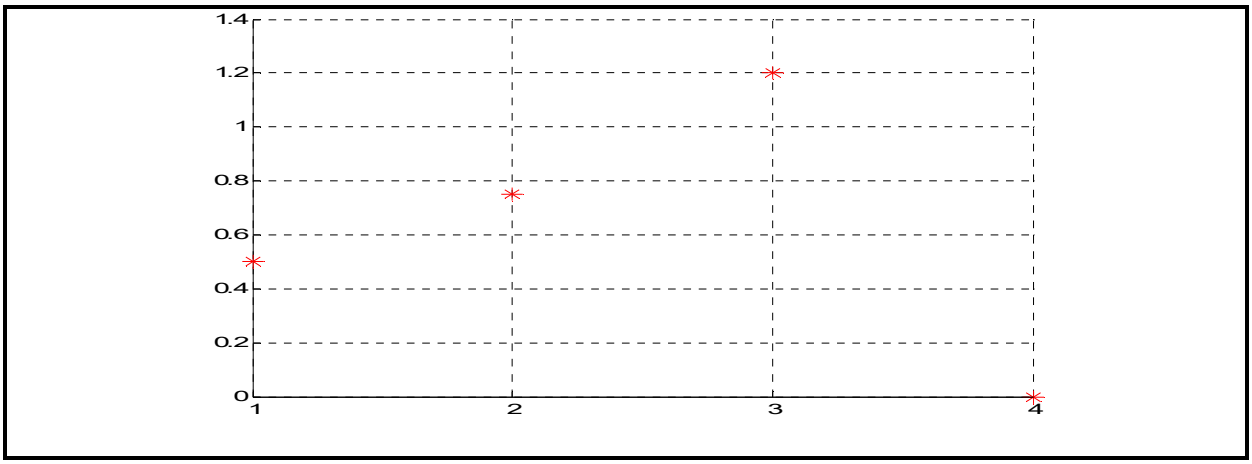


Figure 5.3- Représentation de la réponse impulsionnelle du filtre FIR sur les4 premiers en utilisant la commande «filter »

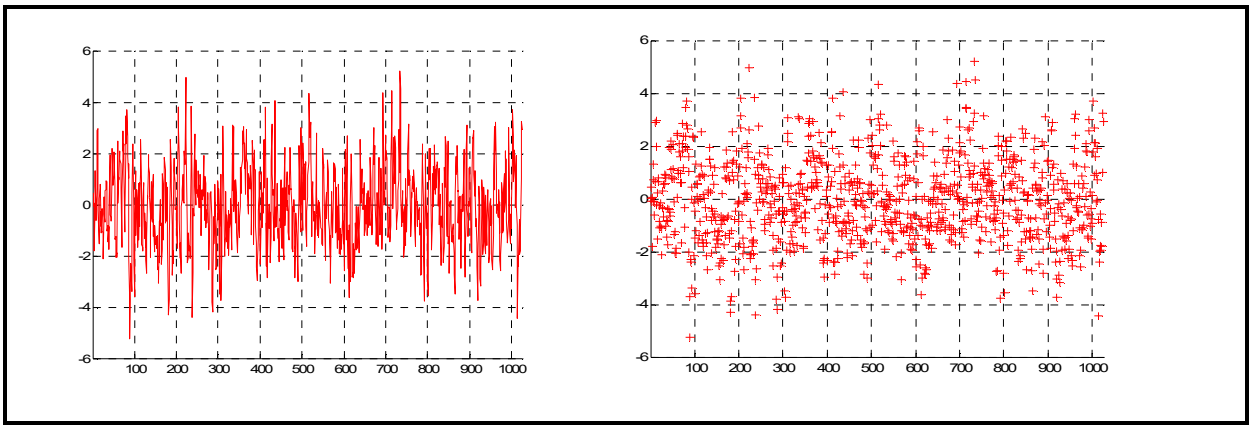


Figure 5.4- **Représentation de la sortie du système ayant comme entrée un bruit blanc**

Annexe

Annexe

Quelques fonctions Matlab utiles dans le TP

`plot`	permet de tracer une fonction
`xlabel`	rajoute une légende à l'axe des abscisses
`ylabel`	rajoute une légende à l'axe des ordonnées
`title`	rajoute un titre à une figure
`axis`	permet de modifier la valeur des axes
`fft`	calcule une transformée de Fourier Rapide
`ifft`	calcule une transformée de Fourier inverse
`linspace(a,b,n)`	génère un vecteur de n valeurs équidistantes entre a et b
`abs`	calcule une valeur absolue ou un module dans le cas complexe
`real`	extrait la partie réelle d'un nombre complexe
`imag`	extrait la partie imaginaire d'un nombre complexe

`fftshift` : recentre le spectre

`psd` : densité spectrale de puissance

Filtrage et convolution	
`conv`	convolution et produit de polynômes
`conv2`	convolution bidimensionnelle
`deconv`	déconvolution et division de polynôme
`filter`	filtrage numérique avec filtre de type RIF (réponse impulsionnelle finie) et avec filtres de type RII (réponse impulsionnelle infinie)
`filter2`	filtrage numérique bidimensionnel

Transformation de Fourier	
`abs`	module d'un nombre complexe
`angle`	argument d'un nombre complexe
`cplxpair`	associe des nombres complexes par paires de nombres deux à deux conjugués
`fft`	transformée de Fourier rapide monodimensionnelle
`fft2`	transformée de Fourier rapide bidimensionnelle
`fftshift`	permet le centrage du spectre d'une transformée de Fourier rapide
`ifft`	transformée de Fourier inverse monodimensionnelle
`ifft2`	transformée de Fourier inverse bidimensionnelle
`nextpow2`	détermine la puisance de 2 immédiatement supérieure au nombre passé en argument
`unwrap`	corrige les angles des phases

B.9 Polynômes et interpolation

Polynômes	
`conv`	convolution et produit de polynômes
`deconv`	déconvolution et division de polynôme
`poly`	détermine un polynôme à partir de ses racines
`polyder`	dérivation polynomiale
`polyeig`	problème polynomial de valeurs propres
`polyfit`	ajustement polynomial au sens des moindres carrés
`polyval`	évaluation d'un polynôme en un point
`polyvalm`	évaluation d'un polynôme de matrice
`residue`	décomposition en élément simples
`roots`	calcul des racines d'un polynômes

Fonctions mathématiques élémentaires	
abs	valeur absolu ou module
angle	argument d'un complexe
sqrt	racine carrée
real	partie réelle
imag	partie imaginaire
conj	complexe conjugué
gcd	PGCD
lcm	PPCM
round	arrondi à l'entier le plus proche
fix	troncature
floor	arrondi vers -∞
ceil	arrondi vers +∞
sign	signe de
rem	reste de la division
exp	exponentiel
log	log népérien
log10	log décimal
Fonctions trigonométriques	
sin, asin, sinh, asinh	
cos, acos, cosh, acosh	
tan, atan, tanh, atanh	
cot, acot, coth, acoth	
sec, asec, sech, asech	1./cos(z), acos(1./z), 1./cosh(z), acosh(1./z)
csc, acsc, csch, acsch	1./sin(z), asin(1./z), 1./sinh(z), asinh(1./z)

Instruction de contrôle	
if	test conditionnel
else	complète if
elseif	complète if
end	terminaison de if, for et while
for	instruction de répétition avec compteur

while	instruction de répétition avec test
break	interrompt une boucle for ou while
return	retour
error	affiche un message et interrompt l'exécution

Traitement du son	
saxis	modification de l'échelle d'amplitude
sound	convertit un vecteur en son
auread	lit un fichier audio au format SUN
auwrite	écritit un fichier audio au format SUN
lin2mu	conversion loi linéaire vers loi μ
mu2lin	conversion loi μ vers loi linéaire

Traitement de signal	
corrcoef	coefficients decorrélation

cov	matrice de covariance
filter	fitrage monodimensionnel
filter2	filtrage bidimensionnel
cplxpair	tri en paires complexes
unwrap	suppression des sauts de phase
nextpow2	puissance de 2 immédiatement supérieure
fft	FFT monodimensionnel (fréquences de 0 à 1)
fft2	FFT bidimensionnel
ifft	FFT inverse
ifft2	FFT inverse
fftshift	FFT (fréquences de -1/2 à 1/2)

Graphiques 2D	
plot	graphe linéaire
loglog	graphe en échelle log-log
semilogx	graphe en échelle semi-log (abscisse)
semilogy	graphe en échelle semi-log (ordonnée)
fill	graphe de polynômes 2D remplis
polar	graphe en coordonnées polaires

bar	histogramme bâtons
stairs	fonction en marches d'escalier
errorbar	graphe avec barres d'erreur
hist	histogramme
rose	histogramme en pseudo camembert
compass	représentation (module,argument) polaire
feather	représentation (module,argument) linéaire
fplot	graphe d'une fonction

Annotation de graphiques	
title	titre du graphique
xlabel	légende abscisse
ylabel	légende ordonnée
zlabel	légende cote
grid	dessin d'une grille
text	texte
gtext	placement de texte avec la souris
ginput	entrée graphique par la souris
Contrôle des fenêtres graphiques	
figure	ouvre une fenêtre graphique
gcf	retourne le numéro de la figure courante
clf	efface la figure courante
close	ferme la figure courante
hold	gère la surimpression
ishold	état de la surimpression
subplot	sous fenêtres graphique
axes	axes en position arbitraire
gca	retourne le numéro des axes courants
axis	contrôle l'apparence et l'échelle des axes
caxis	contrôle l'échelle des axes et de la pseudocouleur
whitebg	dessine sur fond blanc
cinvert	video inverse

Sauvegarde et copie graphique	
print	imprime ou sauve dans un fichier
printopt	configuration de l'imprimante
orient	orientation paysage ou portait

Contrôle des fenêtres graphiques	
figure	ouvre une fenêtre graphique
gcf	retourne le numéro de la figure courante
clf	efface la figure courante
close	ferme la figure courante
hold	gère la surimpression
ishold	état de la surimpression
subplot	sous fenêtres graphique
axes	axes en position arbitraire
gca	retourne le numéro des axes courants
axis	contrôle l'apparence et l'échelle des axes
caxis	contrôle l'échelle des axes et de la pseudocouleur
whitebg	dessine sur fond blanc
cinvert	video inverse

Bibliographie

[1] A. Sondes, cours de traitement de signal, 2010

[2] M. Bellanger, traitement numérique du signal, Edition DUNOD, 1998.

[3] F. Cottet, Traitement des signaux et acquisition de données, Cours et exercices corrigés, 4ième édition, Dunod, Paris, 2015.

[4] A. Sondes, T P de Traitement de Signal, Génération, Corrélation Et Produit De Convolution Des Signaux Continus, 2010.

[5] T. Neffati, Traitement du signal analogique : Cours, Ellipses Marketing, 1999.

[6] A. Sondes, T P de Traitement de Signal, Transformation De Fourier Et Echantillonnage Des Signaux Analogiques, 2010.

[7] M. Benidir, Théorie et traitement du signal : Méthodes de base pour l'analyse et le traitement du signal, Dunod, 2004.

[8] M. Bellanger, Traitement numérique du signal : Théorie et pratique, 9ième édition, Dunod, Paris, 2012.

[9] E. Tisserand, J. Pautex, P. Schweitzer, Analyse et traitement des signaux méthodes et applications au son et à l'image 2ième édition, Dunod, Paris, 2008.

[10] P. Duvaut, F. Michaut, M. Chuc, Introduction au traitement du signal - exercices, corrigés et rappels de cours, Hermes Science Publications, 1996.

[11] Hoang Le-Huy, Introduction à Matlab et Simulink

[12] http://www.sciences.univ-nantes.fr/physique/perso/aloui/matlab/

[13] http://www.mmas.univ-metz.fr/~jmse/tpmatlab/introduction.html