UNIVERSITE ABOU BEKR BELKAID–TLEMCEN

FACULTE DE TECHNOLOGIE

DEPARTEMENT DE TÉLÉCOMMUNICATIONS

POLYCOPIE DE TRAVAUX PRATIQUES

Traitement Numérique du Signal

MASTER 1

Télécommunication / Électronique/ Génie Biomédical

Année Universitaire 2019-2020

Présenté par :

Mr BOUSAHLA Miloud Maitre de Conférences

Mr DERRAZ Foued Maitre de Conférences

Table des matières

Préambule

Ce polycopié rassemble les travaux pratiques du module Traitement Numérique du Signal, il est spécialement destiné aux étudiants en M1 MASTER Système de Télécommunications (ST) et Réseaux et Télécommunication (RT) ainsi qu'aux étudiants dans les filières de Génie Électrique et Électronique (GEE) et Génie Biomédical.

Ces Tps ont été effectués au département de Télécommunications de l'Université Abou-Bekr-Belkaid de Tlemcen durant les trois dernières années. Ces TPs sont dans le programme du master 1 Système de Télécommunications. Tous les TPs sont réalisés avec le logiciel Matlab.

Les séries de TP sont à donner à l'étudiant avant chaque séance, cela l'aidera à bien préparer son TP et lui permettra de ne pas perdre du temps à lire l'énoncé au moment de la séance de TP. Il est important de noter qu'un TP bien préparé est beaucoup enrichissant pour l'étudiant.

TP 1 : Génération de signaux sous Matlab

Objectif du TP

Le traitement du signal consiste à étudier et analyser le signal et extraire les informations pertinentes. La plupart de ces signaux sont par nature analogiques, c'est-à-dire qu'ils sont fonction d'une variable t (le temps) continue, et qu'eux-mêmes varient de manière continue. Ces signaux peuvent être aussi traité et analysé sous forme numérique, à l'aide de convertisseurs analogiques-numériques (échantillonnage, quantification et numérisation des signaux analogiques).

Ce TP est une initiation au traitement du signal sous Matlab. Son objectif est d'apprendre à générer, visualiser et analyser quelques signaux analogiques et échantillonnées, ainsi que de mettre en application les connaissances acquises sur l'échantillonnage, notamment le phénomène de recouvrement (Aliasing).

I. Rappel

I.1 Échantillonnage

I.1.1 Définition

L' échantillonnage est une étape importante dans l'analyse du signal. Cette opération consiste à représenter un signal à temps continu $x(t)$ par ses valeurs $x(n\,Te)$ à des instant multiples de T_e. T_e étant la période d'échantillonnage.

Pour un signal continu (t) , qu'on échantillonne à toutes les T_e secondes, on obtient un signal discret $x_e(t)$ qu'on note $x(n\,Te) = x(n)$.

I.1.2 Echantillonnage idéalisé

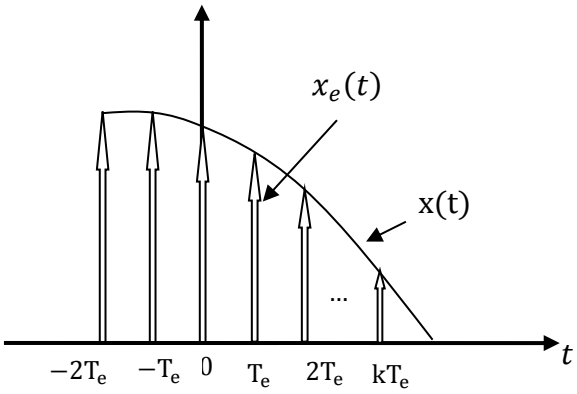
Mathématiquement, échantillonner un signal c'est multiplier ce signal par un peigne de Dirac $\delta_{T_e}(t)$:

$$x_e(t) = x(t)\,\delta_{T_e}(t) = x(t)\sum_{n=-\infty}^{+\infty}\delta(t - n\,T_e)$$

mibousahla@gmail.com

$$x_e(t) = \sum_{n=-\infty}^{+\infty} x(t)\,\delta(t - n\,T_e) \;=\; \sum_{n=-\infty}^{+\infty} x(n\,T_e)\,\delta(t - n\,T_e)$$

Le signal échantillonné est un train d'impulsion de Dirac de période T_e , chaque impulsion ayant comme poids la valeur du signal $x(t)$ d'entrée. C'est donc une distribution.



Modélisation d'un signal échantillonné

I.1.3 Spectre du signal échantillonné

L'opération d'échantillonnage affecte le spectre $X(f)$ du signal.

Notons :

$$X_e(f) = TF[x_e(t)]$$

$$X_e(f) = TF\big[x(t)\,\delta_{T_e}(t)\big] = X(f) * \frac{1}{T_e}\left[\sum_{n=-\infty}^{+\infty} \delta\left(f - \frac{n}{T_e}\right)\right]$$

$$X_e(f) = \frac{1}{T_e}\sum_{n=-\infty}^{+\infty} X(f) * \;\delta\left(f - \frac{n}{T_e}\right)$$

$$X_e(f) = F_e \sum_{n=-\infty}^{+\infty} X(f - n\,F_e)$$

Le spectre de fréquence de $x_e(t)$ est la répétition du spectre de $x(t)$, à la période F_e, multiplié par la constante F_e.

mibousahla@gmail.com

L'échantillonnage a introduit une périodicité dans l'espace des fréquences, ce qui constitue une caractéristique fondamentale des signaux échantillonnés.

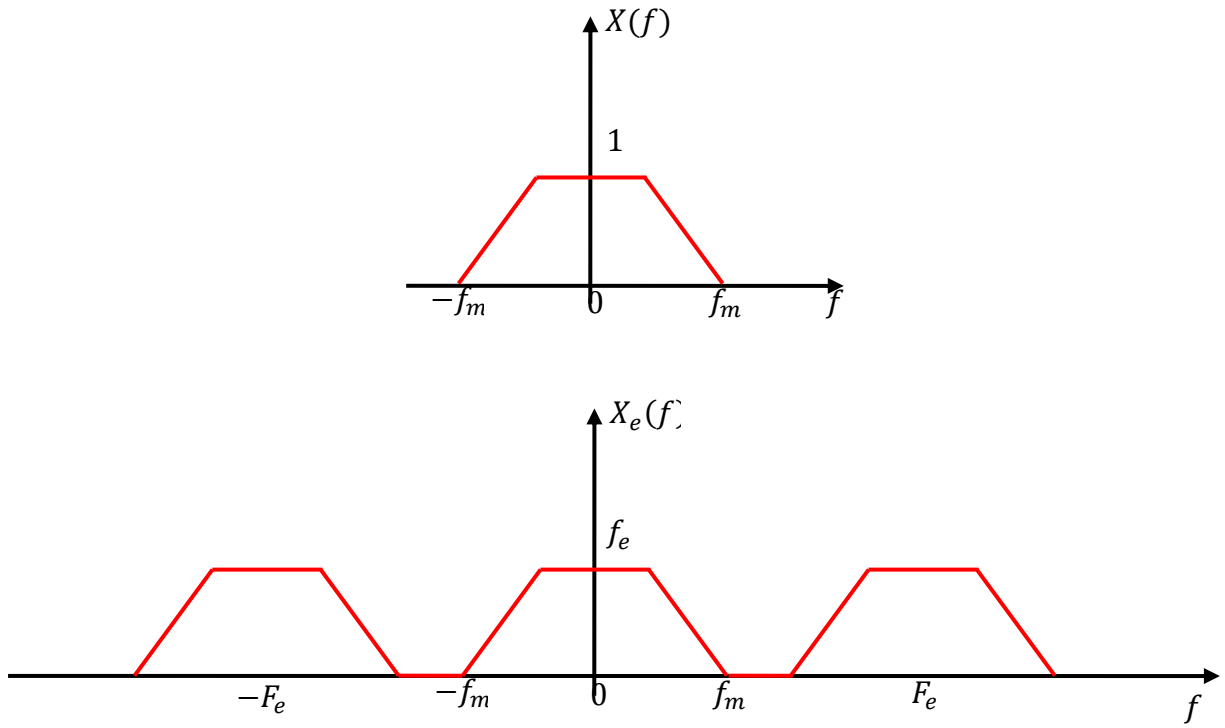


Figure : spectre du signal échantillonné : $F_e \geq 2f_{max}$

I.1.4 Théorème d'échantillonnage

Un signal est correctement représenté par la suite de ces échantillons prélevés avec la période T_e s'il est possible, à partir de cette suite de valeurs de restituer intégralement le signal d'origine.

L'échantillonnage a introduit, dans l'espace des fréquences, une périodicité du spectre. Restituer le signal d'origine, c'est supprimer cette périodicité, c'est-à-dire éliminer les bandes images. Ce traitement peut être réalisé à l'aide d'un filtre passe bas dont la fonction de transfert vaut $\frac{1}{F_e}$ jusqu'à la fréquence $\frac{F_e}{2}$ et 0 aux fréquences supérieures.

mibousahla@gmail.com

Théorème de Shannon :

Pour que la reconstitution d'un signal $x(t)$ à spectre borné à f_{max}, échantillonné à la fréquence F_e ,soit possible par le filtrage passe-bas du signal échantillonné $x_e(t)$, il faut que : $F_e \geq 2f_{max}$.

La condition de Shannon permet d'échantillonner un signal sans aucune perte d'information. Si cette condition n'est pas respectée on observe un "repliement de spectre".

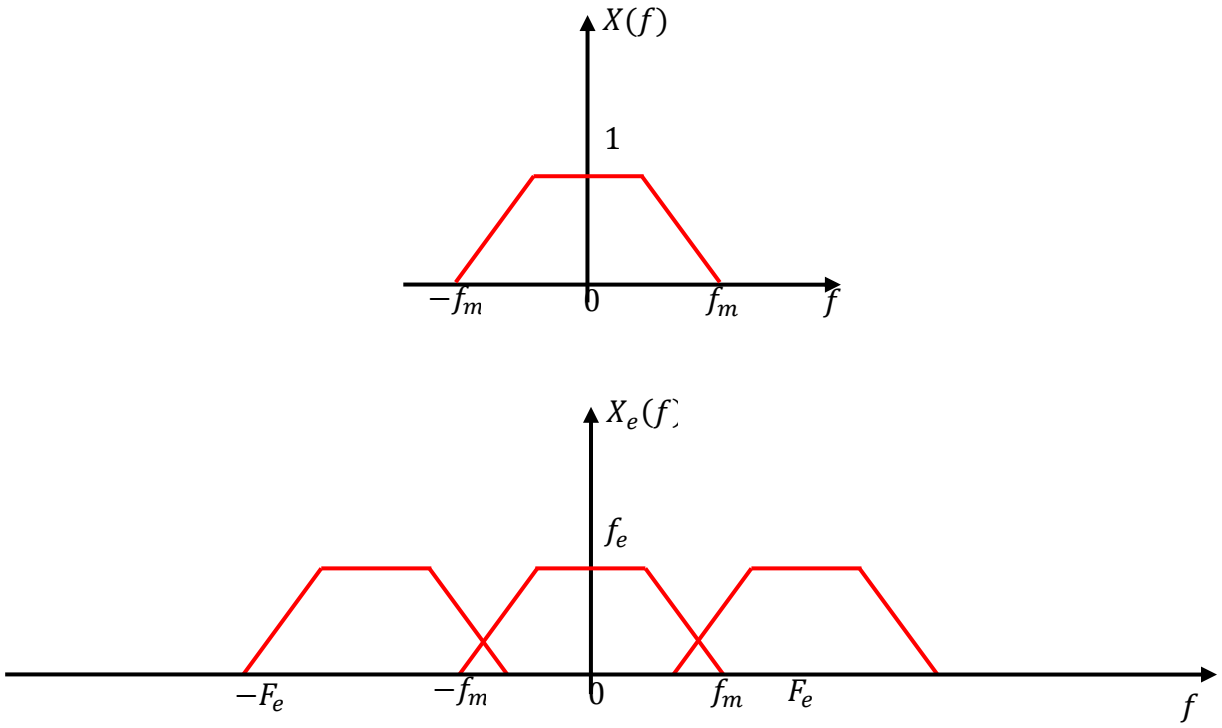


Figure : Repliement de spectre $F_e < 2f_{max}$

I.2 Représentation des signaux et systèmes

Un signal numérique échantillonné à la fréquence Fe se représente dans Matlab, comme un vecteur de N éléments. Sa durée est égale à N*Te et la forme du vecteur des temps t qui lui est associé dans matlab est t=début : pas : duree-pas avec pas= Te et duree =N***Te.**

mibousahla@gmail.com

I.2.1 Axe temporel et fréquentiel pour un signal numérique

Un signal numérique est défini par un nombre d'échantillons **N** relevés à une fréquence d'échantillonnage **Fe**. Les signaux sont toujours captés de manière temporelle, mais souvent on s'intéresse à leur allure fréquentielle.

Il est important de respecter les grandeurs physiques du signal. Il faut donc bien définir les axes temporels et fréquentiels relatifs au signal. Ainsi, les axes sont définis comme suit :

L'axe temporel est un vecteur de N points espacés de $\frac{1}{F_e}$. Sous Matlab, la syntaxe est :

```
t = (1:N)/Fe;
```

L'axe fréquentiel est un vecteur de N points compris entre 0 et F_e . La syntaxe est :

```
f = (0:N-1)*(Fe/N);
```

I.2.2 Opérations sur les signaux

Sous Matlab les signaux doivent être stockés dans un vecteur ligne ou colonne suivant les préférences de l'utilisateur.

Les opérations sur les signaux se font comme des opérations sur les matrices. C'est-à-dire la définition d'un vecteur se fait à l'aide de "**:**". Le pas par défaut est 1, mais on peut définir le pas en l'intercalant entre le début et la fin du vecteur.

X=1:6 donne x= [1 2 3 4 5 6];
pas = 0.5;
x=1:pas:6 donne x= [1 1.5 2 2.5 3 3.5 4 4.5 5 5.5 6];

l'appel d'un élément d'un vecteur se fait en notant le numéro de l'élément entre parenthèse ; Idem pour une matrice (ligne, colonne).

x = 11:-1:1
x = 11 10 9 8 7 6 5 4 3 2 1

x(4)
ans = 8

La somme de 2 vecteurs se fait terme à terme sur des vecteurs de même longueurs. Il est possible de vérifier la taille des vecteurs à l'aide de la commande length.

Le produit par une constante d'un vecteur multiplie tous les termes par cette constante.

(1:N)*5 = 5:5:5*N;

Le produit de 2 vecteurs de même taille renvoie une erreur. Il est possible d'effectuer le produit terme à terme de deux vecteurs. L'opérateur est ".*".

(1:4).*(1:4) = [1 4 9 16];

II. Manipulation

II.1 Génération des signaux : Impulsion Unité et Echelon Unité

Deux signaux élémentaires de base sont : l'impulsion Unité et l'échelon Unité.

L'impulsion unité delta(n) de longueur N peut être générée en utilisant la commande MATLAB suivante :

delta = [1 zeros(1,N-1)] ;

De même, l'impulsion unité deltad(n) de longueur N et décalées de M échantillons, tel que M<N, peut être générée par la commande MATLAB suivante :

deltad = [zeros(1,M) 1 zeros(1,N-M-1)];

L'échelon unité u(n) de longueur N peut être généré en utilisant la commande MATLAB suivante : u=[ones(1,N)]

La version décalée de l'échelon unité s'obtient de la même manière que dans le cas de l'impulsion unité.

Afin de générer l'impulsion unité on peut écrire le programme suivant :

```
%impulsion unité
t=-20:30;
x=[zeros(1,20),1,zeros(1,30)];
stem(t,x);
axis([-20 30 -0.5 1.5]);
title('Impulsion unité');
xlabel('n');
ylabel('Amplitude');
```

1- Tapez et testez le programme précédent. Expliquez à quoi servent les différentes fonctions.

2- Modifier le programme pour générer et tracer une impulsion unité décalée de 11 échantillons et une autre avancée de 8 échantillons.

3- Pour le cas de l'échelon unité, on se contentera d'un nombre fini d'échantillon. Modifier le programme pour générer et tracer un signal échelon unité.

4- Modifier le programme pour générer et tracer un échelon unité décalé de 20 échantillons.

II.2 Génération de signaux sinusoïdaux

Une autre classe de signaux très utile en traitement du signal sont les signaux sinusoïdaux de la forme :

$x(t) = A\ \sin(2\ pi\ F_0 t + \varphi)$

Avec MATLAB, ce type de signaux peut être généré en utilisant les opérateurs trigonométriques cos et sin.

1- Simuler l'échantillonnage à 10 kHz d'une sinusoïde de fréquence f de 500 Hz. Calculer 200 échantillons et visualiser le signal échantillonné en fonction du temps.

Pour afficher un signal sous forme de peigne de Dirac on utilise la commande matlab stem.

Est-ce-qu'on obtient les résultats attendus : amplitude, fréquence, nombre de périodes observées ?

2- Générer et tracer, sur une durée de 1 ms, un signal sinusoïdal $x_1\ (t)$ de fréquence 1000 Hz, échantillonné à 8000 Hz.

3- Sur la même figure, à la même fréquence d'échantillonnage et sur la même durée, générer et tracer un signal sinusoïdal $x_2\ (t)$ de fréquence 9000 Hz, échantillonné à 8000 Hz.

Commenter.

II.3 Génération d'un signal exponentiel

Un autre type de signaux de base sont les signaux exponentiels. Ce type de signaux peut être généré dans matlab par exp et ^ .

Générer sur une durée de 1 ms, un signal exponentiel $x_1\ (t)$ de fréquence 1000 Hz, échantillonné à 8000 Hz. Tracer la partie réelle et imaginaire de ce signal.

$$x_1\ (t) = e^{j*2*pi*F_0*t}$$

mibousahla@gmail.com

Les commandes Matlab suivantes sont utiles pour faire votre travail :

figure : créé une nouvelle figure. On peut rappeler la première figure par **figure(1)**.

subplot(N,M,n) permet de diviser une figure en NxM graphiques. Chaque graphique est désigné par un numéro entre 1 et N.M (de gauche à droite et de haut en bas).

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.

axis : indique les échelles des axes de coordonnées.

clf : efface la figure en cours.

hold on permet de superposer plusieurs courbes sur le même graphique. **hold off** annule cette commande et n'autorise qu'un seul *plot* par graphique.

N= length(x) : renvoie dans N la dimension du vecteur x.

mibousahla@gmail.com

TP N° 2 : Transformée de Fourier Discrète TFD

Objectif du TP

L'algorithme de la transformation de Fourier rapide FFT est un algorithme rapide permettant de calculer la transformée de Fourier discrète (TFD) d'un signal échantillonné. La fonction FFT de Matlab implante cet algorithme de calcul de la transformation de Fourier discrète.

Notre but dans ce TP est de voir qu'il est possible d'estimer la transformée de Fourier d'un signal analogique quelconque (périodique ou pas) à l'aide de la TFD en utilisant un calculateur.

I. Rappel

I.1 Transformée de Fourier

La transformée de Fourier d'un signal analogique $x(t)$est donnée par :

$$X(f) = TF(x(t)) = \int_{-\infty}^{+\infty} x(t) e^{(-j\,2\,\pi\,f\,t)}\, dt$$

I.2 Transformée de Fourier à temps discret (DTFT)

Pour les signaux discrets, la transformée s'appelle : Transformée de Fourier à temps discret (DTFT).

Un signal à temps discret n'est défini qu'aux instants d'échantillonnage nT_e, multiples entiers de la période d'échantillonnage T_e.

La transformée de Fourier d'un signal discret $x_e(t)$ s'écrit :

$$X(F) = \sum_{n=-\infty}^{+\infty} x(n\,T_e) e^{\left(-j\,2\,\pi\,\frac{F}{F_e}\,n\right)} = \sum_{n=-\infty}^{+\infty} x(n) e^{\left(-j\,2\,\pi\,\frac{F}{F_e}\,n\right)}$$

$$X(f) = \sum_{n=-\infty}^{+\infty} x(n) e^{(-j\,2\,\pi\,f\,n)} \;\; avec\; f = \frac{F}{F_e} : fréquence\; réduite\; ou\; normalisée$$

mibousahla@gmail.com

Si on veut calculer la transformée de Fourier X(f) sur ordinateur ou sur un système numérique, on a 2 problèmes :

- La somme est sur un intervalle infini : Le calcul de la TF nécessite une infinité de points (échantillons)
- La variable f est continue : Le calculateur ne peut calculer le contenu fréquentiel du signal discret qu'en un nombre fini L de points fréquentiels, or f varie continûment.

C'est la raison pour laquelle la notion de transformée de Fourier discrète (DFT : Discrete Fourier Transform) a été introduite, pour laquelle on :

- Limite la somme sur un intervalle fini.
- Discrétise la variable f (échantillonnage fréquentiel).

Le calcul de la transformée de Fourier discrète est limité à un nombre fini (N) de points de la suite temporelle $\{x(n)\}$ et à un nombre fini (L) de valeurs de la fréquence f.

$$X(k) = \sum_{n=0}^{N-1} x(n)\, e^{\left(-j\frac{2\pi}{N}nk\right)}\ , \qquad k \in \{0,1,2,\ldots..(N-1)\}$$

L'intérêt pratique de la DFT est très largement dû à la découverte d'une méthode de calcul rapide connue sous le nom de transformée de Fourier rapide (FFT : Fast Fourier Transform).

I.3 Transformée de Fourier rapide FFT :

L'algorithme de la transformation de Fourier rapide FFT est un algorithme rapide permettant de calculer la transformée de Fourier discrète (TFD) d'un signal échantillonné. La fonction FFT de Matlab implante cet algorithme de calcul de cette transformation de Fourier discrète.

Pour calculer la FFT, on introduit les échantillons du signal à la fonction FFT :

$$Xf = fft(x)$$

Si le nombre d'échantillons du signal n'est pas une puissance de deux, on a intérêt à compléter le signal par des échantillons de valeur nulle afin de pouvoir utiliser l'algorithme de la FFT. Cette technique porte le nom de technique de remplissage par des zéros (zero padding en anglais). Le remplissage d'une séquence numérique par des zéros fournit une meilleure représentation de $X(k)$, sans pour autant augmenter la précision du résultat.

La commande $fft(x, Ntfd)$ calcule la transformée de Fourier discrète sur $Ntfd$ points du vecteur x, complété avec des zéros si x a moins de $Ntfd$ points et tronqué si x a plus de $Ntfd$ points.

La FFT calculée est de taille identique que celle du signal traité. Pour le vérifier on utilise la commande whos.

Les spectres calculés sont complexes, les instructions abs et angle permettent de calculer la magnitude et la phase de la transformée de Fourier obtenue :

magXf=abs(Xf)

phaseXf=angle(Xf)

La commande unwrap est utilisée afin d'avoir l'information de phase représentée correctement par rapport au domaine fréquentiel :

phaseXf=unwrap(phaseXf)

La TF et la TF inverse sont des opérateurs à valeurs complexes. La transformée de Fourier Inverse s'obtient par la commande $x = abs(ifft(Xf))$.

II. Manipulation :

II.1 Manip 1

Taper et tester le programme suivant.

Qu'est-ce-qu'il permet de faire ? Commenter.

```
% manip 1
%
clc
clear all
close all
%
N=16;
xn=ones(1,N);
figure(1)
subplot(3,1,1);
stem(xn);
grid
%
Xf1=fft(xn);
Xf1c=fftshift(Xf1);
%
figure(2)
%subplot(3,1,1); plot((0:N-1)/N, abs(Xf1));
%
axe_f=-1/2:1/N:1/2-(1/N);
subplot(3,1,1); plot(axe_f, abs(Xf1c));
grid
legend('Nf=16')
%
xn=[ones(1,N) zeros(1,32-N)];
```

```
%
figure(1)
subplot(3,1,2);
stem(xn);
grid
%
Xf2=fft(xn);
Xf2c=fftshift(Xf2);
%
figure(2)
Nf=length(xn);
axe_f=-1/2:1/Nf:1/2-(1/Nf);
subplot(3,1,2); plot(axe_f,abs(Xf2c));
grid
legend('Nf=32')
%
%
xn=[ones(1,N) zeros(1,128-N)];
figure(1)
subplot(3,1,3); stem(xn); grid
%
Xf3=fft(xn);
Xf3c=fftshift(Xf3);
%
figure(2)
Nf=length(xn);
axe_f=-1/2:1/Nf:1/2-(1/Nf);
subplot(3,1,3); plot(axe_f,abs(Xf3c));
grid
legend('Nf=128')
```

II.2 Manip 2

Soit le signal $x(n) = [1\ 2\ 1\ 0]$.

1. Calculer et tracer le spectre d'amplitude de $x(n)$ en utilisant la commande fft(x).

2. Calculer et tracer le spectre d'amplitude de $x(n)$ en utilisant la commande fft(x, Nfft), avec Nfft=8, 16, 32. Que constatez-vous ?

3. Reprendre la question 1 mais en prenant soin de compléter le signal $x(n)$ avec des valeurs nulles (opération dite de "Zero Padding " avant de calculer la TFD. Commencer avec 4 zéros, puis essayer avec 12 zéros, … etc. Commenter.

4. Quel est l'effet du zéros padding sur la précision de la représentation graphique de $X(f)$.

II.3 Manip 3

Soit le signal exponentiel $x\ (t)$ de fréquence $F = 60\ Hz$.

$$x\ (t) = e^{j*2*pi*F*t}$$

mibousahla@gmail.com

Écrire un script Matlab qui permet de générer ce signal sur une durée de 0.5 s. La fréquence d'échantillonnage est prise égale à 1 kHz.

Compléter le programme pour calculer et tracer le spectre de ce signal en utilisant la commande fft. Commenter.

En utilisant la commande fftshift tracer le spectre de ce signal entre $-F_e/2$ et $+F_e/2$. Commenter.

II.4 Manip 4

1. Soit le signal sinusoïdal à temps continu suivant :

$$x(t) = A\ \cos(2\ pi\ F\ t))$$

Avec : $A = 1.5\ V\ \ ,\ F = 60\ Hz$

Écrire un programme Matlab qui permet de générer et tracer ce signal pour t allant de 0 à 0.5 s. La fréquence d'échantillonnage est égale à 1 kHz.

Compléter le programme pour calculer et tracer le spectre d'amplitude de ce signal en utilisant la commande fft. Commenter.

En utilisant la commande fftshift tracer le spectre d'amplitude de ce signal entre $-F_e/2$ et $+F_e/2$. Commenter.

Fonctions Matlab susceptibles de vous être utiles

fft calcule une transformée de Fourier Rapide

fftshift affiche le spectre dans une bande fréquentielle centrée en 0.

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

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

Rappel : Une aide en ligne de toutes les fonctions Matlab sont disponibles grâce à la commande *: help nom_de_function*

mibousahla@gmail.com

TP N° 3 : Analyse spectrale par Transformée de Fourier Discrète

Objectif :

L'objectif de ce TP est d'étudier dans le détail la représentation fréquentielle des signaux. On analyse l'effet de la troncation (fenêtrage) dans le temps d'un signal sur son spectre et on s'intéresse aux différents paramètres (fenêtre de pondération, fréquence d'échantillonnage et durée d'observation) qui peuvent influencer la qualité de la représentation fréquentielle et temporelle des signaux.

On désire voir dans ce TP différents effets de la TFD en termes de précision d'analyse et de résolution fréquentielle.

I. Rappel

I.1 Résolution fréquentielle

Le principal usage de la TFD est lié à la mise en évidence des caractéristiques spectrales des signaux, le plus souvent analogiques.

La résolution fréquentielle est la capacité à détecter deux fréquences voisines contenues dans le spectre d'un signal. C'est donc le plus petit écart entre deux fréquences détectables. Une bonne résolution fréquentielle est une résolution faible.

Si l'écart en valeur absolue entre deux fréquences voisines F_1 et F_2 dans le spectre du signal est supérieur à F_e/N , il sera possible de distinguer ces deux fréquences sur le tracé.

La résolution fréquentielle est liée au nombre de points du signal et à la forme de la fenêtre de pondération.

$$\text{Résolution fréquentielle} = \frac{F_e}{N}$$

D'une manière générale, plus la durée d'analyse du signal est grande, plus la résolution fréquentielle est meilleure. Par exemple pour un signal périodique, la TFD donne le spectre exact si le signal est analysé sur un multiple entier de sa période, et si la fenêtre d'analyse du

signal est elle-même un multiple entier de la période d'échantillonnage ; sinon le spectre n'est qu'approché.

La résolution fréquentielle ne doit pas être confondue avec la précision qui est liée à la mesure d'une fréquence.

I.2 Fenêtres de pondération

Un signal ne peut être correctement restitué par N valeurs de sa Transformée de Fourier Discrète que si sa durée est inférieure ou égale à NTe. Dans la réalité, les signaux ne sont pas à durée limitée.

L'analyse spectrale d'un signal de longue durée n'utilise, en pratique, qu'une portion limitée de ce signal. Donc le spectre obtenu est égal au spectre du signal à analyser multiplié au préalable par une fenêtre dans le domaine temporel.

Pour ne pas déformer le spectre théorique du signal, il faudrait que le spectre $W(f)$ de la fenêtre de troncature (figure 1) se rapproche le plus possible d'une distribution de Dirac qui est l'élément neutre du produit de convolution. Deux éléments importants sont nécessaires pour se rapprocher de la distribution de Dirac. La finesse du lobe principal et la hauteur des lobes secondaires. Plus la largeur du lobe principal est fine, plus la résolution est grande, donc on peut séparer des fréquences proches. Et plus les niveaux des lobes secondaires sont élevés plus on altère la forme du spectre et la détection d'un signal d'amplitude faible en présence d'un signal d'amplitude élevée sera difficile.

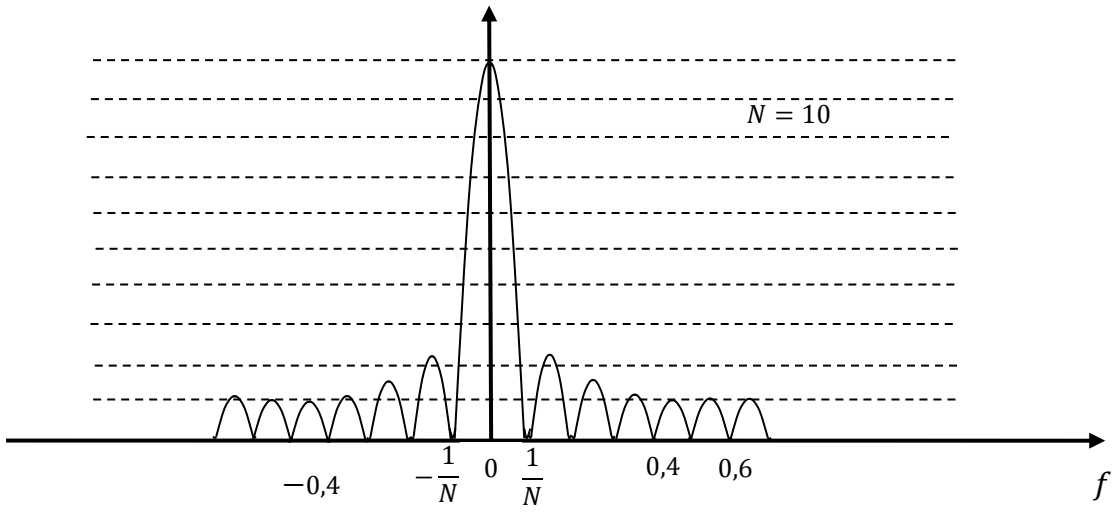


Figure 1 : TFD d'une fenêtre rectangulaire pour N=10

mibousahla@gmail.com

Le but principal des fenêtres de pondération est de limiter les effets de troncature. En effet, le calcul de la TFD par calculateur utilise la technique du zero padding qui consiste à compléter par des échantillons de valeurs nulles pour aller jusqu'à la puissance de 2 la plus proche de la durée choisie. Cette mise à zéro brutale entraîne l'apparition de lobes secondaires qui dans certains cas posent problème lorsque le spectre d'un signal est composé de nombreuses fréquences proches les unes des autres. Une raie d'amplitude faible au voisinage d'une raie d'amplitude plus élevée sera cachée par le premier lobe secondaire.
Malheureusement, la réduction du niveau des lobes secondaires s'accompagne toujours de l'élargissement du lobe principal. Il faut donc accepter un compromis entre ces deux effets.
Un autre problème se pose dans la séparation de raies très voisines mais de même amplitudes, car les 2 raies seront confondues dans un lobe principal plus large.

De nombreuses études ont été faites afin d'obtenir des fenêtres temporelles possédant un spectre plus proche des caractéristiques souhaitées, permettant ainsi de minimiser les déformations. Les fenêtres **triangulaire, Hanning, Hamming, …** permettent de limiter le signal en le pondérant au lieu de le tronquer. Chaque fenêtre réalise une pondération différente des autres. La question du choix de la fenêtre s'impose donc.

I.2.1 Fenêtre Rectangulaire (fenêtre naturelle)

Le fait de n'utiliser que *N* valeurs d'un signal {*x(n)*}, pour le calcul de sa TFD, peut être interprété comme la multiplication de la totalité de ce signal par la suite $w_N(n) = 1_{\{0,1,\dots,N-1\}}(n)$. Cette suite à N échantillons de valeur constante égale à l'unité est appelée fenêtre rectangulaire. Dans le domaine spectral, cette multiplication est équivalente à une convolution de la TFTD de $x(n)$ par la TFTD $W_N(f)$ de la suite $w_N(n)$:

$$\{x(n) \times w_N(n)\} \xrightarrow{TF} X(f) * W_N(f)$$

Où :

$$W_N(f) = \frac{\sin(N\,\pi\,f)}{\sin(\pi\,f)} e^{(-j\ \pi\,(N-1)f\ )}$$

La TFTD $W_N(f)$ de cette fenêtre, qui est un sinus cardinal qui s'annule tous les 1/N, possède un lobe principal de largeur (2/N) et plusieurs lobes secondaires. L'amplitude des lobes secondaires est à peu près 13 en dessous de celle du lobe principal. La largeur du lobe principale (2/N) peut être réglée par le nombre d'échantillons N. Donc, plus la durée d'observation est

longue, plus la résolution est meilleure. Par contre, le niveau des lobes secondaires varie très peu en fonction de N donc toujours une distorsion de spectre.
Le spectre obtenu par Transformée de Fourier Discrète, est donc un ensemble de fonctions sin cardinal centrées sur les fréquences du spectre théorique du signal.

I.2.2 Autres fenêtres

Quelques caractéristiques pour les fenêtres les plus courantes sont données au tableau suivant. A_{dB} est l'atténuation en dB de la hauteur du plus grand lobe par rapport à la hauteur du lobe principal. On constate que la diminution de l'amplitude des lobes secondaires se fait au prix d'une augmentation de la largeur du lobe principal.
Dans un problème d'analyse spectrale, afin d'obtenir un bon compromis résolution/déformation, on utilise généralement plusieurs fenêtres de pondération l'une après l'autre.

Type	**Expression pour $n \in \{0, 1, \dots, N\}$**	**Largeur du lobe principal**	**A_{dB}**
Rectangulaire	$\Pi_N = 1_{\{0,\dots,N-1\}}(n)$	$\frac{2F_e}{N}$	$-13\ dB$
Triangulaire	Triangle de largeur N	$\frac{4F_e}{N}$	$-25\ dB$
Hann	$0.5 - 0.5\ \cos\left(\frac{2\pi n}{N}\right)$	$\frac{4F_e}{N}$	$-31\ dB$
Hamming	$0.54 - 0.46\ \cos\left(\frac{2\pi n}{N}\right)$	$\frac{4F_e}{N}$	$-41\ dB$
Blackman	$0.42 - 0.5\ \cos\left(\frac{2\pi n}{N}\right) + 0.08\ \cos\left(\frac{4\pi n}{N}\right)$	$\frac{6F_e}{N}$	$-61.5\ dB$

mibousahla@gmail.com

II. Manipulation

Manip 1

Pour comparer les propriétés fréquentielles de différentes fenêtres de pondération en termes de largeur du lobe principal et d'atténuation des lobes secondaires, écrire un programme qui permet :

1° d'afficher sur la même figure les représentations temporelles des différentes fenêtres de pondération. Ces fenêtres sont réalisées par les fonctions matlab suivantes : rectwin, triang, hamming, kaiser, bartlett, blackman, hann.

2 ° de calculer les modules des spectres des différentes fenêtres. Normaliser les spectres de façon à ce que pour la fréquence nulle, ils soient tous égaux à 1.

3 ° d'afficher en décibels, dans l'intervalle fréquentiel $[0, 1/2]$, les représentations des modules des spectres des différentes fenêtres. La fréquence d'échantillonnage est prise égale à 1.

4° Pour la fenêtre de Kaiser, essayez plusieurs valeurs du paramètre qui permet de changer l'atténuation du lobe secondaire et la largeur du lobe principal. Commentez.

5° Dans le domaine fréquentiel, relever la largeur du lobe principal en fonction de la longueur de la fenêtre temporelle ainsi que l'atténuation en décibels du lobe secondaire pour les différentes fenêtres de pondérations. Comparez les résultats.

Ramarque : Pour une meilleure représentation du spectre, utiliser la technique du zero padding qui consiste à ajouter des échantillons nuls au signal avant de calculer sa Transformée de Fourier Discrète. Par exemple utiliser une fenêtre de 64 points et allonger sa durée, en ajoutant des zéros, jusqu'`a 1024 points.

Manip 2

Dans cette manipulation on cherche, grâce à une simulation matlab, à analyser les performances de la TFD. Nous nous intéressons à l'approximation apportée par la limitation à un nombre fini d'échantillons, à l'effet de cette troncation dans le temps et à la discrétisation de l'axe fréquentiel sur le spectre du signal.

Nous voulons étudier par simulation le choix du nombre N d'échantillons observées (limitation du nombre d'échantillons) et du nombre L de points de calcul de la TFD (discrétisation de l'axe fréquentiel) pour le calcul du spectre du signal.

1° Ecrire un programme matlab qui permet de générer et de visualiser la suite $x(n)$ obtenue par échantillonnage, à la fréquence $F_e = 8\,kHz$, du signal $x(t) = 2 \times \sin(2 \times \pi \times F_0 \times t)$ avec $F_0 = 1.3\ 10^3\ Hz$.

2° Fixer le nombre L de points de calcul de la TFD à une valeur fixe par exemple 1024. Calculer et tracer le spectre d'amplitude (module de la TFD) en linéaire et en décibel pour différentes valeurs de N, par exemple 128, 256 … pour mettre en évidence que la précision fréquentielle augmente avec la taille la fenêtre.

3° Commentez : la précision obtenue sur F0, la largeur du lobe principal et la différence en dB entre le lobe principal et le lobe secondaire.

4° Pour chaque valeur du nombre de points N, calculer la durée d'observation du signal. Calculer le nombre de période du signal par durée d'observation.

5° Pour quelles valeurs du nombre de points N et L, le calcul de la TFD donne-t-il qu'une seule raie de valeur non nulle ? Calculer dans ce cas le nombre de période du signal par durée d'observation. Conclure.

Manip 3

Pour illustrer et analyser en détail les résultats fournis par la TFD en terme de précision d'analyse et de résolution fréquentielle, nous considérons un signal $x(t)$ constitué de la somme de deux sinusoïdes de même amplitudes $A_1 = A_2 = 1\,V$ et de fréquences respectives $F_1 = 130\,Hz$ et $F_2 = 150\,Hz$.

$$x(t) = A_1 \times \sin(2 \times \pi \times F_1 \times t) + A_2 \times \sin(2 \times \pi \times F_2 \times t)$$

1° En utilisant Matlab, générer et tracer la suite $x(n)$ obtenue par échantillonnage du signal $x(t)$ à la fréquence $F_e = 1\,kHz$.

2° Tracer le spectre d'amplitude $|X(k)|$ (module de la TFD) du signal $x(n)$ en décibel pour des tailles de fenêtres de pondérations de 16, 32, 64 et 128 points. Utiliser les trois fenêtres suivantes : rectangulaire, Hamming, puis Blackman pour mettre en évidence le problème qui peut être posé par l'augmentation de la largeur du lobe principal dans le cas de sinusoïdes de fréquences très rapprochées et de même amplitudes.

Vérifier si les fréquences F_1 et F_2 sont présentent sur les figures du spectre représenté.

Commenter.

3° Refaire le même travail mais en prenant une amplitude $A_2 = 0.01$ pour mettre en évidence le problème qui peut être posé lorsque le spectre d'un signal est composé d'une raie d'amplitude élevée proche d'une autre d'amplitude faible.

TP N°4 : Analyse de filtres numériques RII et RIF

Objectif du TP

Nous allons étudier dans ce TP, l'analyse de filtres numériques à réponses impulsionnelles finies et infinies à l'aide du logiciel Matlab.

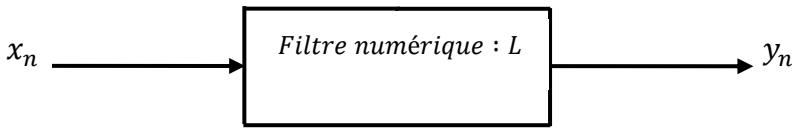
Analyser le comportement d'un filtre numérique dont la fonction de transfert H(z) (ou l'équation de récurrence) est connue c'est déterminer la réponse temporelle et la réponse fréquentielle à une entrée fixée (impulsion, échelon …) ainsi que les caractéristiques de ce filtre.

I. Rappel

I.1 Filtres numériques

Un filtre numérique réalise un calcul sur une séquence de nombres introduite à son entrée et fournit , à sa sortie , une séquence numérique modifiée de sorte à répondre à des contraintes données (réponse en amplitudes , en phase …).

L'outil d'étude des filtres numériques est la transformée en z. Elle permet de modéliser les filtres numériques sous la forme de fonctions de transfert polynômiale, facilitant leur étude.



Le filtrage numérique se pratique sur un système numérique tel qu'un ordinateur (utilisation de logiciels, programmation C/C++.), un Processeur de Traitement de Signal (DSP) ou autre composant numérique (microcontrôleur, FPGA, etc).

I.2 Caractérisation des filtres numériques

Les filtres numériques sont caractérisés par :

- Equation aux différences : relation entrée-sortie dans le domaine temporel.
- Fonction de transfert en Z : H(z)
- Réponse impulsionnelle notée h(n) : la réponse à une entrée impulsion.

I.3 Filtres numériques à réponse impulsionnelle infinie (RII)

Un filtre numérique à réponse impulsionnelle infinie a une rétroaction de la sortie vers l'entrée, et en général sa sortie est fonction des échantillons de sortie précédents et des échantillons d'entrée présents et passés, comme le décrit l'équation suivante :

$$\mathrm{y(m)} = \sum_{k=1}^{N} a_k y(m-k) + \sum_{k=0}^{\mathrm{M}} b_k x(m-k) \qquad (1)$$

En théorie, lorsqu'un filtre RII est excité par une impulsion, la sortie persiste pour toujours. Ainsi, ce type de filtre est connu sous le nom de filtre à réponse impulsionnelle à durée infinie (IIR). Les autres noms pour un filtre RII comprennent les filtres récursif, les filtres à pôle zéro et les filtres à moyenne mobile auto-régressive (ARMA).

La caractéristique du filtre est entièrement déterminée par ses coefficients $\{a_k , b_k\}$, qui sont des constantes calculées pour obtenir une réponse en fréquence spécifiée. Deux filtres RII se distinguent uniquement par les valeurs de leurs coefficients.

La fonction de transfert du filtre, obtenue en prenant la transformée en z de l'équation de différence (1), est donnée par :

$$\mathrm{H(z)} = \frac{\mathrm{Y(z)}}{\mathrm{X(z)}} = \frac{\sum_{k=0}^{\mathrm{M}} b_k z^{-\mathrm{k}}}{1 - \sum_{k=1}^{N} a_k z^{-\mathrm{k}}} \qquad (2)$$

L'ordre du filtre est égal au sup(N,M).

I.4 Filtres à réponse impulsionnelle finie (RIF)

La relation entrée-sortie d'un filtre RIF est donnée par :

$$\mathrm{y(m)} = \sum_{k=0}^{\mathrm{M}} b_k x(m-k) \qquad (3)$$

mibousahla@gmail.com

La sortie y(m) d'un filtre RIF est uniquement fonction du signal d'entrée x(m).

L'ordre du filtre est égal à M.

La réponse d'un tel filtre à une impulsion (réponse impulsionnelle) consiste en une séquence finie de M+1 échantillons. Le filtre est donc connu sous le nom de filtre à réponse impulsionnelle à durée finie (RIF). D'autres noms pour un filtre RIF comprennent le filtre tout zéro, le filtre à action directe ou le filtre à moyenne mobile (MA).

La fonction de transfert du filtre, obtenue en prenant la transformée en z de l'équation de différence (3), est donnée par :

$$\mathrm{H(z)} = \frac{\mathrm{Y(z)}}{\mathrm{X(z)}} = \sum_{k=0}^{\mathrm{M}} b_k \, z^{-\mathrm{k}} \qquad (4)$$

I.5 Réponse fréquentielle d'un filtre numérique

La réponse fréquentielle d'un filtre numérique notée $H\left(e^{j\omega}\right)$ est obtenue à partir de $H(Z)$ en posant :

$$z = e^{j\omega}$$
$$H\left(e^{j\omega}\right) = H(Z)|_{z=e^{j\omega}}$$

Cette réponse fréquentielle est à valeurs complexes. Elle peut donc s'écrire :

$$H\left(e^{j\omega}\right) = \left|H\left(e^{j\omega}\right)\right| e^{j \arg(H(e^{j\omega}))}$$

La réponse d'un filtre numérique est périodique de période égale à 1 en fréquence normalisée. Cette réponse sera tracée pour l'intervalle de fréquence : 0 et ½

I.6 MANIPULATION DES FILTRES SOUS MATLAB

Sous matlab, un filtre numérique est représenté par deux vecteurs contenant les coefficients des puissances de z^{-1}, des polynômes numérateur et dénominateur de sa fonction de transfert en z.

Un filtre RII de fonction de transfert en z $\mathrm{H(z)}$ donnée par :

$$\mathrm{H(z)} = \frac{\mathrm{Y(z)}}{\mathrm{X(z)}} = \frac{\sum_{k=0}^{\mathrm{M}} b_k z^{-\mathrm{k}}}{1 - \sum_{k=1}^{N} a_k z^{-\mathrm{k}}}$$

$\mathrm{H(z)}$ peut aussi s'écrire sous la forme :

$$\mathrm{H(z)} = \frac{\mathrm{Y(z)}}{\mathrm{X(z)}} = \frac{\sum_{k=0}^{\mathrm{M}} b_k z^{-\mathrm{k}}}{1 + \sum_{k=1}^{N} a_k z^{-\mathrm{k}}}$$

sera représentée sous Matlab par :

$Nm = [b_0 \ b_1 \ b_2 \ \cdots \ b_M]$
$Dn = [1 \ a_1 \ a_2 \ \cdots \ a_N]$

Un filtre RIF de fonction de transfert en z $\mathrm{H(z)}$ donnée par :

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

sera représentée sous Matlab par :

$Nm = [b_0 \ b_1 \ b_2 \ \cdots \ b_M]$
$Dn = [1 \ 0 \ 0 \ \cdots \ 0]$

La commande impz permet de calculer la réponse impulsionnelle du filtre.

$[h, n] = impz(Nm, Dn)$

La commande filter permet de calculer la réponse temporelle du filtre à une entrée quelconque.

$yn = filter(Nm, Dn, xn)$

La commande freqz permet de calculer la réponse fréquentielle du filtre.

$[H, f] = freqz(Nm, Dn, N, Fe)$

Où Fe est la fréquence d'échantillonnage et N est le nombre de points de calcul de la réponse fréquentielle entre $\left[0, \frac{F_e}{2}\right]$.

II. TRAVAIL DEMANDÉ SOUS MATLAB

II.1 Écrire le programme Matlab permettant de calculer et de tracer la réponse impulsionnelle (fonction impz), la réponse fréquentielle en module et en phase (fonction freqz), le temps de propagation de groupe (fonction grpdelay) et les pôles et les zéros dans le plan complexe (fonction pzmap et zplane), pour chacun des filtres suivants :

$$\boldsymbol{Filtre\ 1 : y(n) = \frac{x(n) + x(n-1)}{2}}$$

$$\boldsymbol{Filtre\ 2 : \ y(n) = \frac{x(n) - x(n-1)}{2}}$$

$$\boldsymbol{Filtre\ 3 : \ y(n) = 0.1\, x(n-1) + 0.2\, x(n-2) + 0.1\, x(n-3)}$$

$\boldsymbol{Filtre\ 4}$:

mibousahla@gmail.com

$$\boldsymbol{y(n) = 0.1\,x(n) - 0.3\,x(n-2) + 0.5\,x(n-3) - 0.3\,x(n-4) + 0.1\,x(n-6)}$$

$$\boldsymbol{Filtre\ 5 :\ y(n) = x(n) + 0.5\,y(n-1)}$$

$$\boldsymbol{Filtre\ 6 :\ y(n) = x(n) - 0.5\,y(n-1)}$$

$$\boldsymbol{Filtre\ 7 :\ y(n) = x(n-1) + 0.5\,y(n-1)}$$

$$\boldsymbol{Filtre\ 8 :\ y(n) = y(n-1) - 0.25\,y(n-2) + x(n) + x(n-1)}$$

Commenter les résultats obtenus : type de filtre, réponse fréquentielle (passe bas, passe haut, …), phase du filtre, …

II.2 En utilisant la fonction filter de Matlab, calculer et tracer le signal de sortie $y(n)$ du filtre 3 lorsque l'entrée est :

$$x(n) = \begin{cases} 1 & n = 0,1 \\ 0 & ailleurs \end{cases}$$

Calculer théoriquement la sortie du filtre pour $n = 0 \ldots 5$ et vérifier votre calcul avec le résultat donné par la fonction filter.

II.3 Calculer la sortie du filtre 1 en utilisant la fonction filter de Matlab pour un signal sinusoïdal en entrée donné par :

$$x1 = 1.5 * \sin(2 * pi * F1 * t) \quad avec\ F1 = 300\ Hz\ \ et\ \ F_e = 10\ kHz$$

Tracer les deux signaux d'entrée et de sortie du filtre sur la même figure. Commenter.

II.4 Toujours en utilisant le même filtre, calculer et tracer le signal de sortie en utilisant la fonction filter de Matlab pour le signal d'entrée suivant :

$$x = x1 + 0.5 * \sin(2 * pi * F2 * t) \quad avec\ F2 = 4500\ Hz$$

Commenter.

mibousahla@gmail.com

TP 5 : Synthèse de filtres numériques à réponses impulsionnelles infinies RII

Objectif du TP

Nous allons étudier dans ce TP, la synthèse de filtres numériques à réponses impulsionnelles infinie RII à l'aide du logiciel Matlab qui permet d'effectuer la synthèse à la fois de filtres numériques ou analogiques.

Le but de ce TP est de bien comprendre les méthodes de synthèse des filtres RII. Ce TP est constitué d'un rappel de cours sur les filtres numériques à réponses impulsionnelles infinie RII et leurs méthodes de synthèse suivie des différentes commandes matlab permettant la conception de ces filtres.

I. Rappels théoriques

I.1 Filtres numériques à réponses impulsionnelles infinies (RII)

Un filtre numériques à réponse impulsionnelle infinie (RII) a une rétroaction de la sortie vers l'entrée, et en général sa sortie est fonction des échantillons de sortie précédents et des échantillons d'entrée présents et passés, comme le décrit l'équation aux différences suivante :

$$\mathrm{y(m)} = \sum_{k=0}^{\mathrm{M}} b_k\, x(m-k) + \sum_{k=1}^{N} a_k\, y(m-k) \qquad (1)$$

$x(m)$: valeurs successives du signal d'entrée

a_k, b_k : coefficients de la fonction de transfert du filtre,

$y(m)$: valeurs successives du signal de sortie,

En théorie, lorsqu'un filtre RII est excité par une impulsion, la sortie persiste pour toujours. Ainsi, un filtre à réponse impulsionnelle à durée infinie (IIR) est également connu sous le nom de filtre récursif. Les autres noms pour un filtre RII comprennent les filtres à pôle zéro et les filtres à moyenne mobile auto-régressive (ARMA).

La caractéristique du filtre est entièrement déterminée par ses coefficients $\{a_k , b_k\}$. Les coefficients $\{a_k , b_k\}$ sont des constantes calculées pour obtenir une réponse en fréquence spécifiée.

La fonction de transfert du filtre, obtenue en prenant la transformée en z de l'équation de différence (1), est donnée par :

$$\mathrm{H(z)} = \frac{\mathrm{Y(z)}}{\mathrm{X(z)}} = \frac{\sum_{k=0}^{\mathrm{M}} b_k z^{-\mathrm{k}}}{1 - \sum_{k=1}^{N} a_k z^{-\mathrm{k}}} \qquad (2)$$

$$ou \quad H(z) = \frac{\mathrm{Y(z)}}{\mathrm{X(z)}} = \frac{\sum_{i=0}^{N} b_i \, Z^{-i}}{1 + \sum_{i=1}^{M} a_i \, Z^{-i}}$$

Les filtres RII sont caractérisés par une fonction de transfert exprimée par une fraction rationnelle en Z^{-1} :

La réponse en fréquence de ce filtre peut être obtenue à partir de l'équation (2) en posant $z = e^{j\omega}$:

$$\mathrm{H}\left(e^{j\omega}\right) = \frac{\sum_{k=0}^{\mathrm{M}} b_k \, e^{-j\omega k}}{1 - \sum_{k=1}^{N} a_k \, e^{-j\omega k}} \qquad (3)$$

L'ordre du filtre est égal au sup(N,M).

Exemple :

$y(n) = x(n) + a_1 \, y(n-1)$ est l'équation aux différences finies d'un filtre RII.

I.2 Propriétés des filtres RII

Les filtres RII sont très efficaces en temps de calcul. L'introduction de pôles dans H(z) réduit considérablement le nombre de coefficients par rapport à un filtre équivalent RIF.

Leurs inconvénients sont :

- une réponse en phase non linéaire en général.
- une instabilité numérique : le filtre est récursif, les erreurs de précision numérique peuvent s'amplifier. La stabilité des filtres RII doit être vérifiée lors de la conception.

I.3 Synthèse de filtres numériques

Nous avons vu comment analyser un filtre numérique dont la fonction de transfert H(z) ou l'équation aux différences est connue. Nous avons déterminé la réponse temporelle à une entrée fixée (par exemple impulsion, échelon …) et la réponse fréquentielle. Ceci suppose que le filtre est déjà connu. En général, le problème qui se pose est l'inverse, on désire déterminer les valeurs des coefficients du filtre, c'est-à-dire déterminer la fonction de transfert en z ou l'équation aux différences du filtre qui doit avoir une réponse temporelle imposée ou une réponse fréquentielle satisfaisant à un gabarit donné. On dit alors que l'on fait la synthèse du filtre numérique.

La figure 1 représente la réponse fréquentielle d'un filtre passe-bas idéal et d'un filtre passe haut idéal. Dans un cas réel, il est impossible d'obtenir une fréquence de coupure aussi raide et le passage entre bandes passantes et bandes atténuées se fait par des bandes dites "de transition " $f_p - f_a$. Aussi les bandes passantes et atténuées contiennent des ondulations dont l'amplitude est exprimée par les paramètres d'ondulation en bande passante δ_p et bande atténuée δ_a (**figure 2 : filtre passe bas**).

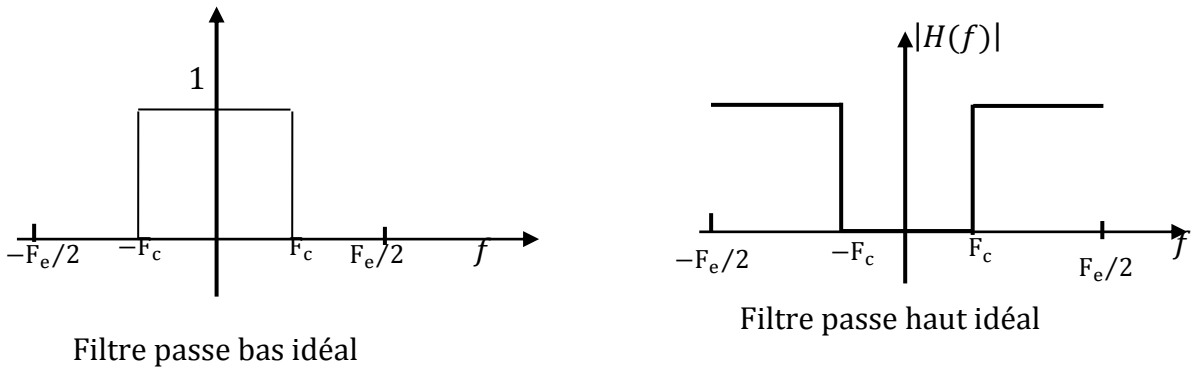
La fréquence de coupure f_C est obtenue pour une atténuation de 3dB.

La bande $[0, f_C]$ est la *bande passante*,

La bande $[f_p,\ f_a]$ est la *bande de transition*,

La bande $[f_a, +\infty]$ est la *bande atténuée*.

Un filtre passe-bas, de fonction de transfert $H(p)$ est normalisé si sa pulsation de coupure est de 1 rd/s et si $|H(0)| = 1$.



Filtre passe bas idéal

Filtre passe haut idéal

Figure 1 : Réponse fréquentiel

mibousahla@gmail.com

La synthèse ou la conception de filtre numérique commence toujours par la spécification du filtre. On utilise souvent un filtre passe bas de référence que l'on adapte. Ce filtre de référence doit être réalisable.

Il faut définir :

- La fréquence de coupure fc désirée (souvent on exprime la fréquence de coupure f_c sous forme normalisée par rapport à la fréquence d'échantillonnage).
- Les valeurs limites de la bande passante fp et bande atténuée fa
- Les valeurs permises des ondulations en bande passante δ_p et atténuée δ_a
- La largeur de la zone de transition Δ_f permise, qui doit être la plus petite possible.
- L'ordre maximal permis.

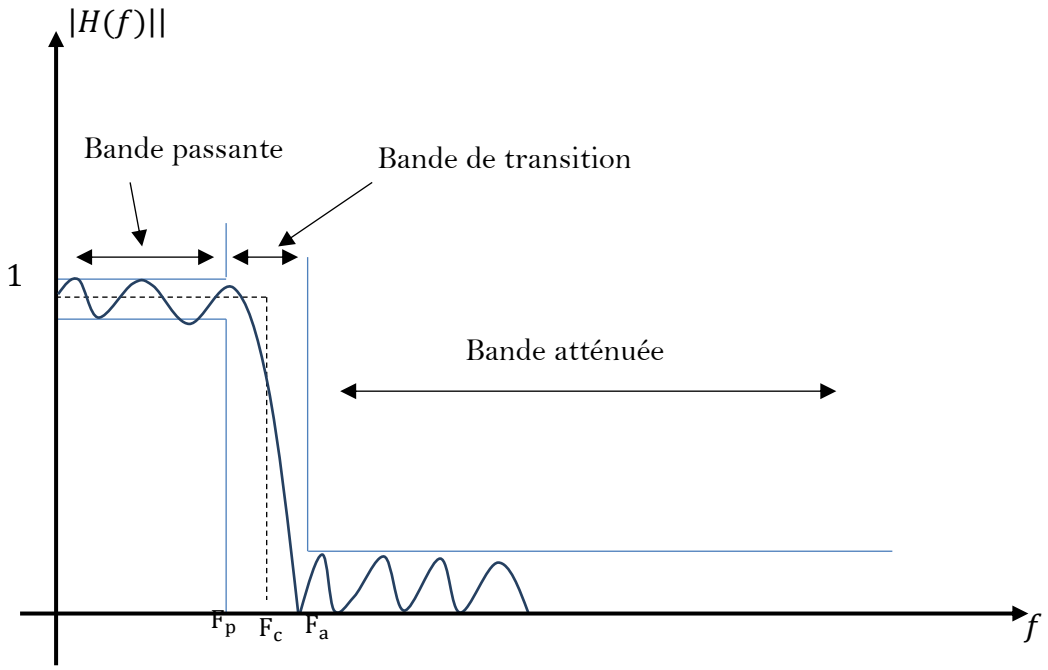


Figure 2 : Réponse fréquentiel désirée, filtre passe bas idéal

Plus f_p et f_a sont proches, plus l'ordre sera élevé

Pour un filtre idéal, $f_p = f_a = f_c$

L'ondulation en bande passante est :

$A_p(dB) = 20\ \log_{10}(1 + \delta_p)$ ou $\delta_p = 10^{0.05A_p} - 1$

L'ondulation en bande atténuée est :

$A_a(dB) = -20\ \log_{10}(\delta_a)$ ou $\delta_a = 10^{-0.05A_a}$

mibousahla@gmail.com

I.4 Méthodes de synthèse de filtres numériques RII

Il existe deux approches permettant de synthétiser un filtre numérique RII :

- Placement manuel des pôles et zéros dans le plan z.
- Concevoir un filtre analogique et le convertir en filtre numérique.

I.4.1 Méthode du placement de pôles et zéros

Cette méthode demande beaucoup de calculs. Elle est basée sur le principe que, dans le plan z :

- Le placement d'un zéros près ou sur le cercle unité dans le plan z minimise la fonction de réponse en fréquence du filtre à cet endroit.
- Le placement d'un pôle près ou sur le cercle unité dans le plan z maximise la fonction de réponse en fréquence du filtre à cet endroit.

Pour obtenir un filtre avec des coefficients réels (donc réalisable), il faut que les pôles et zéros soient à valeurs réelles ou qu'ils apparaissent par paires conjuguées.

I.4.2 Conversion formelle ou transformation d'un filtre analogique de référence

Ces méthodes, dite méthodes indirectes ou de transposition, exploitent le fait qu'il existe des méthodes établies de conception de filtres analogiques. Leur principe consiste à concevoir un filtre analogique de référence, en utilisant des méthodes de synthèse des filtres analogiques aboutissant à une fonction $H(p)$ correspondant aux spécifications, et à le convertir en filtre numérique en utilisant une fonction qui permet le passage du plan p dans le plan z (p = fct(z)). Cette fonction doit maintenir la stabilité du filtre analogique et maintenir les caractéristiques de la réponse fréquentielle $H(f)$ du filtre numérique.

Parmi ces méthodes on peut citer :

- les méthodes de l'invariance impulsionnelle et de l'invariance indicielle : le filtre doit avoir une réponse impulsionnelle ou indicielle imposée.
- la transformation bilinéaire (équivalence de l'intégration) : le filtre doit avoir une réponse fréquentielle entrant dans un gabarit donné.
- Équivalence de la dérivation (Transformation de d'Euler)
- ….etc

Toutes ces méthodes conservent la stabilité des filtres analogiques, une fois convertis en filtres numériques.

Parmi les filtres analogiques de référence les plus utilisés dans ce cas nous citons :

- Filtres de Butterworth
- Filtres de Tchebychev I et II
- Filtres elliptiques ou de Cauer
- …etc

En transformant les filtres analogiques (filtres de Butterworth, de Tchebychev I et II …) en filtres numériques, on arrive à des fonctions de transfert de type RII.
Dans le cas de la transformation d'un filtre analogique, on part souvent de l'équation d'un filtre passe-bas normalisé que l'on adapte au type désiré (passe haut, passe bande, etc.) avant la conversion.

I.4.2.1 Transformation (Synthèse) par invariance impulsionnelle

Nous pouvons passer de la fonction de transfert en p à la fonction de transfert en z, et donc à l'équation aux différences que nous pouvons programmer, en échantillonnant la réponse impulsionnelle. Donc la réponse impulsionnelle du filtre numérique sera identique à la réponse impulsionnelle du filtre analogique échantillonnée.

- on détermine la réponse impulsionnelle désirée $h_a(t)$ d'un filtre analogique connu.
- on échantillonne cette réponse impulsionnelle à la fréquence F_e et on déduit la suite $h(n)$.
- on recherche la fonction de transfert $H(z)$ du filtre numérique qui a pour réponse impulsionnelle la suite $h(n)$.

Remarque :
L'échantillonnage de la RI périodise le spectre du filtre analogique. Or, le spectre d'un filtre passe-haut n'est pas limité dans les hautes fréquences ; il y a donc chevauchement des spectres. Cette méthode de transformation n'est donc pas adaptée aux filtres passe-haut et passe-bande à bande large.

I.4.2.2 Transformation (Synthèse) par invariance indicielle

Le principe de cette méthode est semblable au cas précédent :

* on détermine la réponse indicielle désirée $y_{ind}(t)$.
* on échantillonne cette réponse indicielle à la fréquence F_e et on déduit la suite $y(n)$.

* on recherche la fonction de transfert H(z) du filtre numérique qui a la même réponse indicielle.

I.4.2.3 Transformation bilinéaire (équivalence de l'intégration)

Cette méthode fait partie des méthodes de transformation d'un filtre analogique de référence en un filtre numérique. Nous utilisons donc la méthode de transformation bilinéaire couplée avec une méthode de synthèse de filtres analogiques (Butterworth, Chebychev, Chebychev inverse, Cauer, ...).

Dans la plupart des cas réels, la fonction de transfert du filtre analogique de référence n'est pas connue ; c'est un gabarit qui est utilisé. La technique consiste alors à :

- synthétiser le filtre analogique à l'aide d'une des méthodes de synthèse de filtres analogiques (Butterworth, Chebychev, Chebychev inverse, Cauer, ...).
- puis d'obtenir le filtre numérique en utilisant la transformation bilinéaire.

La variable z étant définie par $z = e^{p\,T_e}$, le passage du domaine continu (variable p) au domaine discret (variable z) peut s'effectuer en remplaçant p par $p = {\ln z}/{T_e}$ mais cette transformation ne conduit pas à une forme polynômiale. Il faut donc rechercher des méthodes d'approximation pour trouver une transformation qui conserve le quotient de deux polynômes (contraintes matérielles).

La transformation bilinéaire définit une relation entre p et z pour transformer une fonction de transfert analogique en équivalent numérique.

$$\mathrm{p} = \lambda\left(\frac{\mathrm{z}-1}{\mathrm{z}+1}\right) = \lambda\left(\frac{1-\mathrm{z}^{-1}}{1+\mathrm{z}^{-1}}\right) \quad avec \quad \lambda = \frac{2}{T_e}$$

$$\mathrm{H(z)} = H(p)\big|_{p=\frac{2}{T_e}\left(\frac{1-\mathrm{z}^{-1}}{1+\mathrm{z}^{-1}}\right)}$$

Cette transformation sauvegarde les principales caractéristiques du filtre analogique de référence dans le filtre numérique réalisé, notamment la stabilité. Néanmoins, elle introduit une certaine déformation sous forme de compression des fréquences entre le filtre analogique de référence et le filtre numérique réalisé.

Les fréquences dans les domaines p et z ne sont pas les mêmes.

$$f_a = \frac{1}{\pi T_e} tan(\pi \ F_{num} \ T_e)$$

Il est donc nécessaire de pré-déformer le gabarit du filtre analogique pour obtenir le filtre numérique désiré.

I.5 Synthèse de filtres numériques RII à partir des filtres analogiques avec Matlab

I.5.1 Evaluation de l'ordre des filtres de Butterworth, Tchebychev et elliptique

Les fonctions Matlab buttord, cheb1ord et ellipord permettent de calculer l'ordre minimal nécessaire pour la conception d'un filtre numérique ou analogique, passe-bas ou passe bande, satisfaisant à un ensemble de spécifications de conception de filtre : une ondulation dans la bande passante ne dépassant pas Rp dB et une atténuation d'au moins Rs dB dans la bande d'atténuation.

Elles prennent en paramètres :

- Pulsation de fin de bande passante : Wp
- Pulsation dans la bande atténuée (coupée) : Ws
– Atténuation Δ1 maximale (en dB) dans la bande passante : Rp
– Atténuation Δ2 minimale (en dB) dans la bande coupée : Rs

Elles retournent l'ordre du filtre (N) et la pulsation de coupure (Wn).

Les arguments de sortie N et Wn seront utilisées par la suite dans les fonctions matlab butter, cheby1, cheby2 et ellip.

Wn fréquence propre du filtre numérique. Pour un filtre passe-bas Wp et Ws sont les fréquences hautes de la bande passante et basse de la bande coupée. Pour un filtre passe-bande, Wp contient les fréquences basse et haute de la bande passante et Ws les fréquences haute et basse de la bande coupée.

I.5.1.1 Calcul de l'ordre du filtre analogique par les fonctions Matlab

[N, Wn] = buttord (Wp, Ws, Rp, Rs,'s')

[N, Wn] = cheb1ord (Wp, Ws, Rp, Rs,'s')

[N, Wn] = ellipord (Wp, Ws, Rp, Rs,'s')

Dans ce cas seulement, on specifie les fréquences Wp et Ws en radians par seconde.

I.5.1.2 Calcul de l'ordre du filtre numérique par les fonctions Matlab

[N, Wn] = buttord(Wp,Ws,Rp,Rs) ;

[N, Wn] = cheb1ord(Wp,Ws,Rp,Rs) ;

[N, Wn] = ellipord(Wp,Ws,Rp,Rs) ;

Dans ce cas, il faut normaliser les fréquences par rapport à Fe/2.

Exemple :

[N, Wn] = buttord (2*pi*fa1, 2*pi *fa2, delta1, delta2, 's') : Calcul l'ordre minimal N et les fréquences de coupure Wn pour un filtre de Butterworth analogique.

[N, Wn] = buttord (fn1/ (Fe/2), fn2/ (Fe/2), delta1, delta2) : Calcul l'ordre minimal N et les fréquences de coupure Wn pour un filtre de Butterworth numérique.

I.5.2 Synthèse de filtres analogiques passe-bas

I.5.2.1 Filtres analogiques de Butterworth

[z,p,k] = buttap(N) renvoie les pôles et le gain d'un prototype de filtre passe-bas analogique normalisé (pulsation de coupure unité) d'ordre N de Butterworth. La fonction renvoie les pôles dans le vecteur colonne p de longueur N et le gain dans le scalaire k. z est une matrice vide car il n'y a pas de zéros. La fonction de transfert est :

$$H(p) = \frac{K}{(p-p_1)(p-p_2)\cdots(p-p_N)}$$

Les filtres de Butterworth sont caractérisés par une réponse en amplitude qui est au maximum plate dans la bande passante et globalement monotone. Dans le cas du filtre passe-bas, les premières dérivées 2N-1 de la réponse d'amplitude au carré sont nulles à $\omega = 0$. La fonction de réponse d'amplitude au carré est :

$$|H(\omega)|^2 = \frac{1}{1+\left(\frac{\omega}{\omega_0}\right)^{2N}}$$

correspondant à une fonction de transfert avec des pôles équidistants autour d'un cercle dans le demi-plan gauche. La réponse en amplitude à la fréquence angulaire de coupure ω_0 est toujours de $\frac{1}{\sqrt{2}}$ quel que soit l'ordre du filtre. buttap met ω_0 à 1 pour un résultat normalisé.

I.5.2.2 Filtres analogiques de Chebychev

I.5.2.2.1 Filtres analogiques de Chebychev de type I

[z,p,k] = cheb1ap(n,Rp) renvoie les pôles et le gain d'un prototype de filtre passe-bas analogique normalisé de Tchebyshev de type I d'ordre n avec Rp dB d'ondulation dans la bande passante. La fonction renvoie les pôles dans le vecteur colonne p de longueur n et le gain dans le scalaire k. z est une matrice vide, car il n'y a pas de zéros. La fonction de transfert est

$$H(p) = \frac{K}{(p-p_1)(p-p_2)\cdots(p-p_N)}$$

Les filtres de Tchebyshev de type I sont équi-ripple dans la bande passante et monotones dans la bande d'atténuation. Les pôles sont régulièrement espacés autour d'une ellipse dans le demi-plan gauche. La fréquence angulaire ω_0, du bord de la bande passante du filtre de Tchebyshev type I, est fixée à 1, pour un résultat normalisé. C'est la fréquence à laquelle la bande passante se termine et le filtre a une réponse en amplitude de $10^{-\frac{R_P}{20}}$.

I.5.2.2.2 Filtres analogiques de Chebychev de type 2

[z,p,k] = cheb2ap(n,Rs) trouve les zéros, les pôles et le gain d'un prototype de filtre passe-bas analogique normalisé de Tchebyshev de type II d'ordre n avec une ondulation dans la bande d'atténuation Rs dB en dessous de la valeur de crête de la bande passante. cheb2ap renvoie les zéros et les pôles dans les vecteurs colonnes z et p de longueur n et le gain en scalaire k. Si n est impair, z est de longueur n-1. La fonction de transfert est

$$H(p) = K\,\frac{(p-z_1)(p-z_2)\cdots(p-z_N)}{(p-p_1)(p-p_2)\cdots(p-p_N)}$$

Les filtres de Tchebyshev de type II sont monotones dans la bande passante et équiripple dans la bande d'atténuation. La position des pôles est l'inverse de celle de Tcheb1ap, dont les pôles sont régulièrement espacés autour d'une ellipse dans le demi-plan gauche.

La fréquence angulaire ω_0, du bord de la bande d'atténuation du filtre de Tchebychev de type II, est fixée à 1 pour un résultat normalisé. C'est la fréquence à laquelle la bande d'atténuation commence et le filtre a une réponse d'amplitude de $10^{-\frac{R_S}{20}}$.

I.5.2.3 Filtres analogiques elliptiques

[z,p,k] = ellipap(n,Rp,Rs) renvoie les zéros, les pôles et le gain d'un prototype de filtre passe-bas analogique normalisé elliptique d'ordre n, avec Rp dB d'ondulation dans la bande passante et une ondulation dans la bande d'atténuation de Rs dB en dessous de la valeur de crête de la bande passante. Les zéros et les pôles sont renvoyés sous forme de vecteurs colonnes z et p de longueur n et le gain sous forme scalaire k. Si n est impair, z est de longueur n - 1. La fonction de transfert sous forme de pôle zéro factorisé est

$$H(p) = K\,\frac{(p-z_1)(p-z_2)\cdots(p-z_N)}{(p-p_1)(p-p_2)\cdots(p-p_N)}$$

Les filtres elliptiques présentent des caractéristiques de chute plus prononcées que les filtres de Butterworth et de Tchebyshev, mais ils sont caractérisés par une ondulation de même amplitude dans la bande passante et dans la bande d'atténuation. Parmi les quatre types de filtres classiques, les filtres elliptiques répondent généralement à un ensemble donné de spécifications de performance de filtre avec l'ordre de filtrage le plus bas.

ellipap fixe la fréquence angulaire du bord de la bande passante ω_0 du filtre elliptique à 1 pour un résultat normalisé. La fréquence angulaire du bord de la bande passante est la fréquence à laquelle la bande passante se termine et le filtre a une réponse en amplitude de $10^{-\frac{R_P}{20}}$.

I.5.3 Représentation des filtres analogiques en termes de numérateurs et dénominateurs de leur fonction de transfert

[b, a] = zp2tf(z,p,k) : forme une fonction de transfert rationnelle (rapport de deux polynômes) à partir des zéros, des pôles et des gains d'un filtre sous forme factorisée.

Les vecteurs numérateur b et dénominateur a sont renvoyés avec des coefficients de numérateur et de dénominateur en puissances décroissantes de p.

$$H(p) = \frac{B(p)}{A(p)} = \frac{b_1\,p^{n-1} + \cdots b_{n-1}\;p + \;b_n}{a_1\,p^{n-1} + \cdots a_{n-1}\;p + \;a_n}$$

I.5.4 Transformation des filtres passe-bas en tout type de filtres

La transformation est une étape dans le processus de conception du filtre numérique pour les fonctions butter, cheby1, cheby2 et ellip.

I.5.4.1 Transformation de filtre analogique passe-bas en passe-bas $\left(p \rightarrow \frac{p}{\omega_0}\right)$

[bt,at] = lp2lp(b,a, ω_0)

La fonction matlab lp2lp transforme un prototype de filtre analogique passe-bas avec une pulsation angulaire de coupure de 1 rad/s en un filtre analogique passe-bas avec n'importe quelle pulsation angulaire de coupure ω_0 (rad/sec) spécifiée. Les vecteurs lignes b et a spécifient les coefficients du numérateur et du dénominateur du prototype de filtre en puissances décroissantes de p.

lp2lp renvoie le filtre transformé en fréquence dans les vecteurs lignes bt et at (les coefficients du numérateur et du dénominateur en puissances décroissantes de p).

I.5.4.2 Transformation de filtre analogique passe-bas en passe-haut $\left(p \rightarrow \frac{\omega_0}{p}\right)$

[bt, at] = lp2hp(b,a, ω_0) ;

La fonction matlab lp2hp transforme un prototype de filtre analogique passe-bas avec une pulsation angulaire de coupure de 1 rad/s en un filtre analogique passe-haut avec une pulsation angulaire de coupure ω_0 (rad/sec) désirée. Les vecteurs lignes b et a spécifient les coefficients du numérateur et du dénominateur du prototype de filtre en puissances décroissantes de p.

lp2hp renvoie le filtre transformé en fréquence dans les vecteurs lignes bt et at (les coefficients du numérateur et du dénominateur en puissances décroissantes de p).

I.5.4.3 Transformation de filtre analogique passe-bas en passe-bande $\left(p \rightarrow \frac{1}{B}\left(\frac{p}{\omega_0} + \frac{\omega_0}{p}\right)\right)$

[bt, at] = lp2bp(b,a, ω_0, BW) ;

La fonction matlab lp2bp transforme un prototype de filtre analogique passe-bas avec une pulsation angulaire de coupure de 1 rad/s en un filtre analogique passe-bande avec la pulsation centrale et la largeur de bande souhaitées.

mibousahla@gmail.com

Les vecteurs lignes b et a spécifient les coefficients du numérateur et du dénominateur du prototype de filtre en puissances décroissantes de p.

Les scalaires ω_0 et BW spécifient la pulsation centrale et la bande passante en rad/s. Pour un filtre avec $\omega_1 < \omega_2 \rightarrow BW = \omega_2 - \omega_1$ et $\omega_0 = \sqrt{\omega_1\,\omega_2}$

lp2bp renvoie le filtre transformé en fréquence dans les vecteurs lignes bt et at (les coefficients du numérateur et du dénominateur en puissances décroissantes de p).

I.5.4.4 Transformation de filtre analogique passe-bas en coupe-bande

$\left(p \rightarrow B\left(\frac{1}{\frac{p}{\omega_0}+\frac{\omega_0}{p}}\right)\right)$

[bt, at] = lp2bs(b,a, ω_0, BW) ;

La fonction matlab lp2bs transforme un prototype de filtre analogique passe-bas avec une pulsation angulaire de coupure de 1 rad/s en un filtre analogique stop-bande avec la pulsation centrale et la largeur de bande souhaitées.

Les vecteurs lignes b et a spécifient les coefficients du numérateur et du dénominateur du prototype de filtre en puissances décroissantes de p.

Les scalaires ω_0 et BW spécifient la pulsation centrale et la bande passante en rad/s. Si ω_1 est la pulsation basse de coupure et ω_2 la pulsation haute de coupure, alors la pulsation propre du filtre ω_0 et la largeur de bande du filtre sont donnés par : $BW = \omega_2 - \omega_1$ et $\omega_0 = \sqrt{\omega_1\,\omega_2}$

lp2bp renvoie le filtre transformé en fréquence dans les vecteurs lignes bt et at (les coefficients du numérateur et du dénominateur en puissances décroissantes de p).

I.5.5 Synthèse (conception) des filtres

Conception d'un filtre de Butterworth

La fonction matlab **butter** permet la conception d'un filtre de Butterworth numérique ou analogique.

Conception de filtres de Butterworth numériques

[B,A] = butter(N,Wn) conçoit un filtre de Butterworth numérique passe-bas d'ordre N et renvoie les coefficients du filtre dans les vecteurs B (numérateur) et A (dénominateur) de longueur N+1. Les coefficients sont énumérés par puissances décroissantes de z. La pulsation de coupure normalisée Wn doit être de 0,0 < Wn < 1, 1 correspondant à la moitié de la fréquence d'échantillonnage.

mibousahla@gmail.com

Si Wn est un vecteur à deux élémlents, Wn = [W1 W2], butter renvoie un filtre numérique passe-bande d'ordre 2N avec la bande passante W1 < W < W2.

[B,A] = butter(N,Wn, "high") conçoit un filtre numérique passe-haut.

[B,A] = butter(N,Wn, "low") conçoit un filtre numérique passe-bas.

[B,A] = butter(N,Wn, 'stop') est un filtre numérique coupe-bande si Wn = [W1 W2].

Lorsqu'il est utilisé avec trois arguments de gauche, comme dans

[Z,P,K] = butter (...) conçoit un filtre de Butterworth numérique et renvoie les zéros et les pôles dans les vecteurs colonnes Z et P de longueur N, et le gain dans le scalaire K. Cette syntaxe peut inclure n'importe lequel des arguments d'entrée des syntaxes précédentes.

Conception de filtres de Butterworth analogiques

[B,A] = butter(n,Wn, 's') ou [Z,P,K] = butter (n,Wn, 's')

[B,A] = butter(N,Wn, 'high', 's') ou [Z,P,K] = butter(N,Wn, 'high', 's')

[B,A] = butter(N,Wn, 'stop', 's') ou [Z,P,K] = butter(N,Wn, 'stop', 's')

Conçoient respectivement un filtre de Butterworth analogique passe-bas, passe-haut et coupe-bande, avec une fréquence angulaire de coupure Wn en [rad/s] qui peut être supérieure à 1,0.

Remarque :

En général, on utilise la syntaxe [z,p,k] pour concevoir des filtres IIR. Pour analyser ou implémenter le filtre, nous pouvons alors utiliser [z,p,k] avec zp2sos. Si on conçoit le filtre en utilisant la syntaxe [b,a], on pourrait rencontrer des problèmes numériques. Ces problèmes sont dus à des erreurs d'arrondi et peuvent se produire pour n aussi bas que 4.

butter utilise un algorithme en cinq étapes :

1. Il trouve les pôles, les zéros et le gain du prototype analogique passe-bas en utilisant la fonction buttap.

2. Il convertit les pôles, les zéros et le gain en espace d'état.

3. Si nécessaire, il utilise une transformation d'état-espace pour convertir le filtre passe-bas en un filtre passe-bande, passe-haut ou coupe-bande avec les contraintes de fréquence souhaitées.

4. Pour la conception de filtres numériques, il utilise une transformation bilinéaire pour convertir le filtre analogique en un filtre numérique par une transformation bilinéaire avec pré-distorsion de la fréquence. Un réglage minutieux de la fréquence permet au filtre

analogique et au filtre numérique d'avoir la même réponse fréquentielle en amplitude à Wn ou à w1 et w2.

5. Il permet de reconvertir le filtre d'espace d'état à sa fonction de transfert ou à sa forme de gain, pôle et zéro, selon les besoins.

I.5.6 Discrétisation des filtres analogiques

Il existe deux techniques dans matlab qui permettent de discrétiser un filtre analogique et donc d'obtenir les coefficients du filtre numérique à partir de ceux du filtre analogique.

– Transformation d'un filtre analogique en numérique par invariance de la réponse impulsionnelle

[bz, az] = impinvar(b, a, fe) ;

La fonction impinvar crée un filtre numérique dont la réponse impulsionnelle est égale à la réponse impulsionnelle échantillonnée à fe du filtre analogique avec les coefficients b et a, mis à l'échelle par 1/fe. Si l'argument fe est omis, ou s'il est spécifié comme vecteur vide [], il est pris par défaut égal à 1 Hz.

Les vecteurs lignes b et a spécifient les coefficients du numérateur et du dénominateur du filtre analogique en puissances décroissantes de p. fe est la fréquence d'échantillonnage.

impinvar renvoie le filtre numérique dans les vecteurs lignes bz et az.

– Transformation d'un filtre analogique en numérique par transformation bilinéaire

La transformation bilinéaire transforme les filtres analogiques, conçus à l'aide de techniques classiques de conception de filtres, en leurs équivalents discrets.

```
[zd,pd,kd] = bilinear(z,p,k,fe)
[zd,pd,kd] = bilinear(z,p,k,fe,fp)
```

convertit la fonction de transfert du domaine p, spécifiée par z, p et k en son équivalent discret. Les entrées z et p sont des vecteurs colonnes contenant les zéros et les pôles, k est un gain scalaire et fe est la fréquence d'échantillonnage en hertz. bilinear renvoie l'équivalent discret dans les vecteurs colonnes zd et pd et kd scalaire. La fréquence optionnelle fp est en hertz et est utilisée pour la pré-distorsion.

```
[numd,dend] = bilinear(num,den,fe)
[numd,dend] = bilinear(num,den,fe,fp)
```

convertit une fonction de transfert du domaine p, donnée par num et den en son équivalent discret.

Les vecteurs lignes num et den spécifient les coefficients du numérateur et du dénominateur du filtre analogique en puissances décroissantes de p et fe est la fréquence d'échantillonnage. bilinear renvoie le filtre numérique dans les vecteurs lignes numd et dend. La fréquence optionnelle fp est en hertz et est utilisée pour la pré-distorsion.

II. Travail à réaliser sur Matlab

On désire réaliser un filtre passe-bas numérique équivalent aux filtres analogiques de :

- Butterworth,
- Tchebychev type 1 et 2
- Elliptique

Le gabarit du filtre numérique que nous souhaitons obtenir est défini comme suit :

fn1 = 300 Hz : fin *bande passante*

fn2 = 400 Hz : *début bande atténuée*

$\Delta 1$ = 1 dB : *atténuation*

$\Delta 2$ = 40 dB : *atténuation*

Fe = 1000 Hz : fréquence *d'échantillonnage*

II.1 Ecrire le script matlab permettant de :

- Trouver l'ordre minimal requis pour les différentes approximations : Butterworth (buttord), Tchebyshev type 1 et 2 (cheb1ord et cheb2ord) et filtre elliptique (ellipord)
- Calculer les quatre fonctions de transfert numériques (en z) par la méthode de la transformation bilinéaire à l'aide des fonctions cheby1, cheby2, ellip et butter convenablement paramétrées.
- Visualiser sur le même système d'axes les quatre réponses en amplitude.
- Pour vérifier si les filtres conçus sont stables, tracez dans le plan complexe les pôles et les zéros des filtres obtenus par chaque approximation (Butterworth, Tchebyshev…). Commenter.

II.2 Refaire le même travail en utilisant la méthode de synthèse de l'invariance de la réponse impulsionnelle.

- Tracez sur la même figure les réponses impulsionnelles du filtre analogique et son équivalent discret pour chaque type de filtres : Butterworth, Tchebychev type 1 et 2 et Elliptique.

Commentez les résultats.

TP 6 : Synthèse de filtres numériques à réponses impulsionnelles finies RIF

Objectif du TP

Nous allons étudier dans ce TP, la synthèse de filtres numériques à réponses impulsionnelles finie à l'aide du logiciel Matlab qui permet d'effectuer la synthèse à la fois de filtres numériques ou analogiques.

Le but de ce TP est de bien comprendre les méthodes de synthèse des filtres RIF et d'étudier l'influence des paramètres tels que l'ordre du filtre et les fenêtres de troncature ou de pondération sur les caractéristiques de ces filtres.

Ce TP est constitué d'un rappel de cours sur les filtres numériques à réponses impulsionnelles finie RIF et leurs méthodes de synthèse suivie des différentes commandes matlab permettant la conception de ces filtres.

I. Rappels théoriques

I.1 Filtres à réponse impulsionnelle finie (RIF)

Les filtres numériques à réponse impulsionnelle de durée finie (RIF), appelés aussi filtres transversaux, sont un exemple de systèmes discrets linéaires et invariants. Ils ont le plus souvent une structure non récursive.

La relation entrée-sortie d'un filtre RIF est donnée par :

$$y(n) = \sum_{k=0}^{M} b_k \, x(n-k) \qquad (1)$$

La sortie y(n) d'un filtre RIF est uniquement fonction du signal d'entrée x(n).

L'ordre du filtre est égal à M.

Les coefficients b_k constituent la réponse impulsionnelle du filtre (ils sont souvent notés h_k).

mibousahla@gmail.com

La réponse d'un tel filtre à une impulsion (réponse impulsionnelle) consiste en une séquence finie de M+1 échantillons. Le filtre est donc connu sous le nom de filtre à réponse impulsionnelle à durée finie (RIF). D'autres noms pour un filtre RIF comprennent le filtre tout zéro, le filtre à action directe ou le filtre à moyenne mobile (MA).

La fonction de transfert du filtre, obtenue en prenant la transformée en z de l'équation de différence (1), est donnée par :

$$\mathrm{H(z)} = \frac{\mathrm{Y(z)}}{\mathrm{X(z)}} = \sum_{k=0}^{\mathrm{M}} b_k \, z^{-\mathrm{k}} \qquad (2)$$

I.2 Propriétés

- Stabilité inconditionnelle : Les filtres à réponse impulsionnelle finie sont toujours stables car ils admettent des pôles seulement à l'origine (pas de terme de récurrence dans l'équation aux différences).
- Phase linéaire : Les filtres RIF peuvent générer des filtres à phase linéaire
- Ils permettent d'obtenir des filtres à partir d'une réponse en fréquence idéale.
- Approximation : Toute fonction de filtrage numérique stable et causale peut être approchée par la fonction de transfert d'un filtre RIF

I.3 Synthèse de filtres RIF

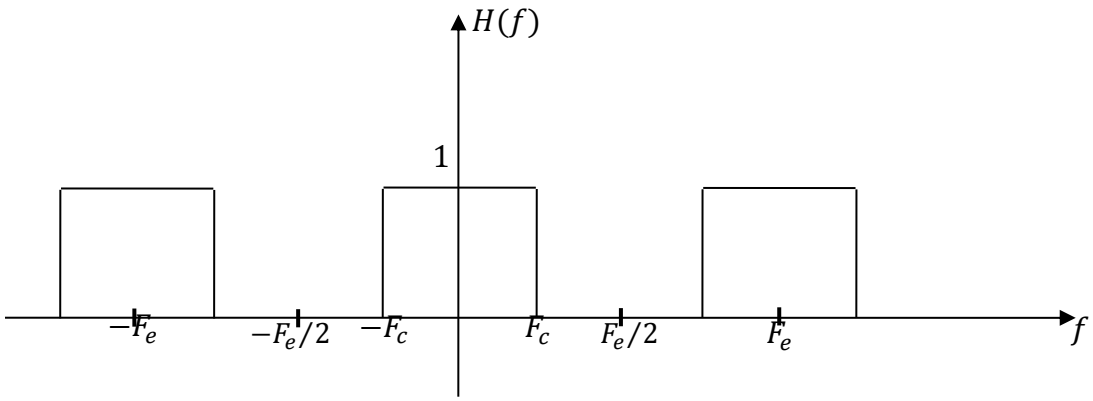
La synthèse des filtres à RIF permet de fixer les valeurs des coefficients de la réponse impulsionnelle. Ces échantillons, appelés coefficients du filtre, sont obtenus en essayant de se rapprocher le plus possible d'une réponse fréquentielle idéale.

À la différence des filtres RII, les filtres RIF ne sont réalisables que dans le domaine discret. Par conséquent, leurs méthodes de synthèse ne sont pas dérivées des filtres analogiques.

I.4 Principe général de la synthèse des filtres RIF

Le principe de la synthèse des filtres RIF est de partir de la réponse en fréquence désirée. Puisqu'on est dans le domaine de l'échantillonné, ces réponses sont périodiques, de période F_e

Par exemple, dans le cas d'un passe-bas :



Réponse fréquentielle d'un filtre passe − bas numérique idéal

1. La méthode de la fenêtre (ou encore de la série de Fourier) : dans cette méthode la synthèse se fait par TF inverse, ou par décomposition en série de Fourier de la réponse fréquentielle : on a la réponse impulsionnelle du filtre continu : Echantillonnage temporel.

2. La méthode de l'échantillonnage fréquentiel : dans cette méthode la synthèse se fait par Transformée de Fourier Discrète inverse depuis un gabarit fréquentiel du filtre continu : Echantillonnage fréquentiel.

3. Les méthodes d'optimisation se concentrent sur la minimisation d'un critère d'erreur entre la courbe réelle et le filtre idéal. La plus utilisée est la méthode de Parks and McClellan, qui reformule le problème de synthèse de filtre sous la forme d'une approximation polynômiale.

I.4.1 Synthèse des filtres RIF par la méthode de la fenêtre (ou encore de la série de Fourier) :

Cette méthode est basée sur l'analyse de Fourier et l'utilisation de fenêtres temporelles. Le plus souvent elle est utilisée pour obtenir un filtre à phase linéaire. Cette propriété est liée à la symétrie des coefficients du filtre donc il est important que la fenêtre de troncature conserve cette symétrie.

On peut réaliser n'importe quel filtre puisqu'il suffit d'imaginer sa réponse fréquentielle $H_a(f)$ puis par transformée de Fourier inverse, de calculer sa réponse impulsionnelle $h_a(t)$ qui fournit, par échantillonnage, la réponse impulsionnelle numérique $h(n)$.

Deux problèmes se posent :

- La suite des échantillons $h(n)$ obtenue est infinie,
- la réponse impulsionnelle n'est pas causale : on ne peut pas réaliser pratiquement ce filtre.

Donc on doit limiter le nombre d'échantillons de la réponse impulsionnelle par troncature de $h(n)$ pour remédier au premier problème. Cette troncature s'obtient en faisant le produit de la réponse impulsionnelle échantillonnée par une séquence de durée finie $w(n)$ appelée fenêtre de pondération. Pour obtenir un filtre RIF de longueur N, on peut par exemple utiliser une fenêtre rectangulaire de largeur N.

Avec cette opération de troncature, on observe une bande de transition entre la bande passante et la bande coupée (la pondération temporelle limite la raideur de coupure du filtre). Cette transition est appelée **transition de Gibbs**. On observe également des ondulations dans les bandes passante et coupée.

Il existe d'autres types de fenêtres de pondération qui ont pour effet de réduire le phénomène de Gibbs. Les principales fenêtres utilisées sont triangulaire, Hanning, Hamming, Blackman.

Le choix d'un type particulier de fenêtre doit être guidé par l'application visée et les performances souhaitées.

Pour obtenir un filtre causal, il faut décaler la réponse impulsionnelle tronquée du nombre d'échantillons d'indice négatif.

I.4.2 Synthèse des filtres RIF par la méthode d'échantillonnage en fréquence

Lorsque l'on ne connaît pas l'expression analytique de $H_a(f)$, la méthode précédente n'est plus applicable puisqu'on ne peut déterminer $h_a(t)$ par transformation de Fourier inverse. On utilise alors la transformation de **Fourier Discrète inverse**. C'est-à-dire que l'on "échantillonne" la réponse désirée dans le domaine fréquentiel, on obtient N points de cette réponse fréquentielle auxquels on fait correspondre N points de la réponse temporelle équivalente obtenus par TFD inverse (ou FFT inverse).

Si l'expression analytique de $H_a(f)$ n'est pas connue, la méthode de la fenêtre n'est plus applicable car on ne peut pas déterminer $h_a(t)$ par transformation de Fourier inverse. On utilise alors la méthode d'échantillonnage en fréquence. Alors on échantillonne la réponse désirée dans le domaine fréquentiel, on obtient donc N points de cette réponse fréquentielle

auxquels on fait correspondre N points de la réponse temporelle équivalente obtenus par TFD inverse (ou FFT inverse).

$$H(k) = H(f)|_{f=\frac{k}{N}} \qquad k = -\frac{(N-1)}{2} \text{ à } \frac{(N-1)}{2}$$

Puis on détermine $h(n)$ par Transformée de Fourier Discrète inverse (TFD inverse).

$$h(n) = \frac{1}{N} \sum_{-\frac{(N-1)}{2}}^{\frac{(N-1)}{2}} H(k)\, e^{\frac{j\,2\,\pi k\, n}{N}}$$

La méthode de synthèse par échantillonnage fréquentiel permet de réaliser toute forme de filtre, chose qu'on ne peut réaliser avec la méthode de la fenêtre. Mais, cette méthode ne garantit que les points fréquentiels $H(k)$. Entre ces points, la valeur de $H(f)$ n'est pas maitrisée, il peut y avoir des oscillations.

I.5 Synthèse de filtres RIF avec Matlab

I.5.1 Synthèse de filtres RIF par la méthode de la fenêtre sous Matlab

fir1 : Conception d'un filtre RIF à phase linéaire par la méthode des fenêtres.

b = fir1(N,Wn) conçoit un filtre numérique RIF d'ordre N à phase linéaire de type passe-bas, passe-bande ou multibande et renvoie les coefficients du filtre dans le vecteur b de longueur N+1. Les coefficients du filtre sont classés par ordre décroissant de décalage :

$$\mathrm{y(n)} = b_0 x(n) + b_1 x(n-1) + b_2 x(n-2) + \cdots + b_N x(n-N)$$

Le type de filtre dépend du nombre d'éléments de Wn.

La fréquence de coupure Wn doit être comprise entre 0 < Wn < 1,0, avec 1,0 correspondant à la moitié de la fréquence d'échantillonnage ou π rad/échantillon. Le filtre b est réel.

Si Wn est un scalaire, alors fir1 conçoit un filtre passe-bas ou passe-haut avec une fréquence de coupure Wn. La fréquence de coupure est la fréquence à laquelle le gain normalisé du filtre est de -6 dB.

Si Wn est le vecteur à deux éléments [w1 w2], où w1 < w2, alors fir1 conçoit un filtre passe-bande ou coupe-bande avec une fréquence de coupure inférieure w1 et une fréquence de coupure supérieure w2.

b=fir1(N,Wn,ftype) : conçoit un filtre passe-bas, passe-haut, passe-bande, coupe-bande ou multibande, selon la valeur de ftype et le nombre d'éléments de Wn :

b = fir1(N,Wn, 'low') conçoit également un filtre passe-bas d'ordre N.

b = fir1(N,Wn, 'high ') conçoit un filtre passe-haut d'ordre N.

Si Wn est un vecteur à deux éléments, Wn = [W1 W2], fir1 conçoit un filtre passe-bande d'ordre N avec une bande passante : W1 < W < W2.
b=fir1(N,Wn, 'bandpass') conçoit également un filtre passe-bande d'ordre N avec une bande passante : W1 < W < W2.
Si Wn = [W1 W2], b = fir1(N,Wn, 'stop') conçoit un filtre coupe-bande.

Si Wn est un vecteur multi-éléments, Wn = [W1 W2 W3 W4 W5 ... WN],
fir1 synthétise un filtre d'ordre N multi bandes à bandes :
0 < W < W1, W1 < W < W2, ..., WN < W < 1.
b = fir1(N,Wn, 'DC-1') fait de la première bande une bande passante.
b= fir1(N,Wn, 'DC-0') fait de la première bande une bande rejetée.

b = fir1(N,Wn,WIN) conçoit un filtre FIR d'ordre N utilisant le vecteur de longueur N+1 WIN pour fenêtrer la réponse impulsionnelle. S'il est vide ou omis, fir1 utilise une fenêtre de Hamming de longueur N+1.

La procédure fir1 utilise par défaut une fenêtre de Hamming. Les autres fenêtres disponibles sont : rectangulaire (rectwin), Hanning, Bartlett, Blackman, Kaiser et Chebwin.
KAISER et CHEBWIN peuvent être spécifiés avec un argument de suivi facultatif. Par exemple, b= fir1(N,Wn,kaiser(N+1,4)) utilise une fenêtre Kaiser avec bêta=4.
b = fir1(N,Wn, 'high',chebwin(N+1,R)) utilise une fenêtre de Tchebyshev avec R décibels d'atténuation relative des lobes latéraux.
Pour les filtres ayant un gain différent de zéro à Fs/2, par exemple les filtres passe-haut et coupe-bande, N doit être pair. Sinon, N sera incrémenté de un. Dans ce cas, la longueur de la fenêtre doit être spécifiée comme N+2.

Exemple :

```
num=fir1(44,0.23,'high',hamming(45)); % numérateur du filtre RIF
den=1; % dénominateur du filtre RIF
freqz(num,den,512) % pour visualiser la réponse fréquentielle de ce filtre
```

Par défaut, le filtre est mis à l'échelle de sorte que le centre de la première bande passante ait une magnitude exactement égale à un après le fenêtrage. On utilise l'argument "noscale" pour empêcher cette mise à l'échelle, par exemple b = fir1(N,Wn, "noscale"),
b = fir1(N,Wn, "high", "noscale"),

b = fir1(N,Wn,wind, "noscale"). Vous pouvez également spécifier explicitement l'échelle, par exemple fir1(N,Wn, 'scale'), etc.

Ordre de filtre : c'est un scalaire entier. Pour les configurations passe-haut et coupe-bande, fir1 utilise toujours un ordre de filtrage pair. L'ordre doit être pair car les filtres FIR symétriques d'ordre impair doivent avoir un gain nul à la fréquence de Nyquist. Si vous spécifiez un n impair pour un filtre passe-haut ou coupe-bande, alors fir1 incrémente N de 1.

Example :

num=fir1(54,0.33,'low',hamming(55)); % numérateur du filtre RIF
den=1; % dénominateur du filtre RIF
freqz(num,den,512) % pour visualiser la réponse fréquentielle de ce filtre

I.5.2 Synthèse de filtres RIF par la méthode de l'échantillonnage de la Réponse en Fréquence

fir2 : La commande fir2 conçoit des filtres numériques RIF à phase linéaire, avec une réponse de forme arbitraire, par la méthode de l'échantillonnage en fréquence.
b = fir2(N,F,A) conçoit un filtre numérique FIR à phase linéaire d'ordre N avec la réponse en fréquence spécifiée par les vecteurs F et A et renvoie les coefficients du filtre en longueur N+1 dans le vecteur b. (cette fonction utilise la transformée de Fourier inverse et une fenêtre de Hamming pour obtenir les coefficients du filtre).
b : est un vecteur ligne contenant les (N+1) coefficients du filtre FIR d'ordre N dont les caractéristiques amplitude fréquence sont données par les vecteurs F et A.
F : est un vecteur de points de fréquence, spécifié dans la plage normalisée $0 \leq F \leq 1$, qui correspond à la limite de la fréquence numérique $0 \leq \omega \leq 1$.
Les fréquences dans F doivent être données en ordre croissant avec $0,0 < F < 1,0$ et 1,0 correspondant à la moitié de la fréquence d'échantillonnage. Le premier et dernier élément de F doivent être égaux à 0 et 1 respectivement.
A : est un vecteur contenant la réponse en amplitude désirée aux points spécifiés dans le vecteur f.

```
f = [0 0.48 0.48 1];
mhi = [0 0 1 1];
bhi = fir2(34,f,mhi); % conçoit un filtre RIF passe-haut d'ordre 34
freqz(bhi,1)
```

Par défaut, fir2 fait le fenêtrage de la réponse impulsionnelle avec une fenêtre de Hamming.

D'autres fenêtres sont disponibles, notamment rectwin, Hann, Bartlett, Blackman, Kaiser et Chebwin. Kaiser et Chebwin peuvent être spécifiés avec un argument de suivi optionnel.

Example :

```
b = fir2(N,F,A,bartlett(N+1))  % utilise une fenêtre de  Bartlett.
b = fir2(N,F,A,chebwin(N+1,R)) % utilise une fenêtre de  Chebyshev.
```

% Example 2 :

```
%   Design a 30th-order lowpass filter and overplot the desired
%   frequency response with the actual frequency response.
 f = [0 0.6 0.6 1];      % Frequency breakpoints
m = [1 1 0 0];          % Magnitude breakpoints
b = fir2(30,f,m);       % Frequency sampling-based FIR filter design
[h,w] = freqz(b,1,128); % Frequency response of filter
plot(f,m,w/pi,abs(h))
legend('Ideal','fir2 Designed')
title ('Comparison of Frequency Response Magnitudes')
```

I.5.3 Synthèse de filtres RIF par les méthodes d'optimisation

I.5.3.1 Synthèse de filtres RIF par la méthode des moindres Carrés

firls conçoit un filtre FIR à phase linéaire qui minimise l'erreur quadratique pondérée et intégrée entre une fonction linéaire idéale par morceaux et la réponse en amplitude du filtre sur un ensemble de bandes de fréquences souhaitées.

b = firls(n,f,a) retourne le vecteur ligne b contenant les n+1 coefficients du filtre FIR d'ordre n dont les caractéristiques fréquence-amplitude correspondent approximativement à celles données par les vecteurs f et a. Les coefficients du filtre de sortie dans b obéissent à la relation de symétrie.

b(k)=b(n+2-k), k=1,...,n+1

Il s'agit de filtres à phase linéaire de type I (n impair) et de type II (n pair). Les vecteurs f et a spécifient les caractéristiques fréquence-amplitude du filtre :

f est un vecteur de paires de points de fréquence, spécifié dans la plage comprise entre 0 et 1, où 1 correspond à la fréquence de Nyquist. Les fréquences doivent être en ordre croissant. Les points de fréquence dupliqués sont autorisés et, en fait, peuvent être utilisés pour concevoir

un filtre exactement identique à ceux renvoyés par les fonctions fir1 et fir2 avec une fenêtre rectangulaire (rectwin).

a est un vecteur contenant l'amplitude souhaitée aux points spécifiés dans f.

La fonction d'amplitude désirée aux fréquences entre les paires de points (f(k), f(k+1)) pour k impair est le segment de droite reliant les points (f(k), a(k)) et (f(k+1), a(k+1)).

La fonction d'amplitude souhaitée aux fréquences entre les paires de points (f(k), f(k+1)) pour k pair n'est pas spécifiée. Il s'agit de régions de transition ou de régions "indifférentes".

f et a sont de même longueur. Cette longueur doit être un nombre pair.

firls utilise toujours un ordre de filtrage pair pour les configurations avec une bande passante à la fréquence de Nyquist. En effet, pour les ordres impairs, la réponse en fréquence à la fréquence de Nyquist est nécessairement 0. Si on spécifie un n impair, firls l'incrémente de 1.

Example :

Conception d'un filtre passe-bas d'ordre 55 avec une bande de transition entre 0.25π et 0.35π.

b = firls(55,[0 0.25 0.35 1],[1 1 0 0]);

I.5.3.2 Synthèse de filtres RIF par la méthode de Parks-McClellan

firpm conçoit un filtre FIR à phase linéaire utilisant l'algorithme de Parks-McClellan. L'algorithme de Parks-McClellan utilise l'algorithme d'échange de Remez et la théorie d'approximation de Tchebyshev pour concevoir des filtres avec un ajustement optimal entre les réponses en fréquence souhaitées et réelles. Les filtres sont optimaux en ce sens que l'erreur maximale entre la réponse en fréquence désirée et la réponse en fréquence réelle est minimisée. Les filtres conçus de cette manière présentent un comportement d'équi-répartition dans leurs réponses en fréquence et sont parfois appelés filtres d'équi-répartition.

firpm présente des discontinuités à la tête et à la fin de sa réponse impulsionnelle en raison de cette nature d'équi-ondulation.

b = firpm(n,f,a) renvoie le vecteur ligne b contenant les n+1 coefficients du filtre FIR d'ordre n dont les caractéristiques fréquence-amplitude correspondent à celles données par les vecteurs f et a.

Les coefficients du filtre de sortie dans b obéissent à la relation de symétrie :

b(k)=b(n+2-k), k=1,...,n+1

Les vecteurs f et a spécifient les caractéristiques de fréquence-amplitude du filtre :

f est un vecteur de paires de points de fréquence normalisée, spécifiée dans la plage comprise entre 0 et 1, où 1 correspond à la fréquence de Nyquist. Les fréquences doivent être en ordre croissant.

a est un vecteur contenant les amplitudes souhaitées aux points spécifiés dans f.
L'amplitude désirée aux fréquences entre des paires de points (f(k), f(k+1)) pour k impair est le segment de droite reliant les points (f(k), a(k)) et (f(k+1), a(k+1)).
L'amplitude souhaitée aux fréquences entre les paires de points (f(k), f(k+1)) pour k pair n'est pas spécifiée. Les zones entre ces points sont des régions de transition ou des régions "indifférentes".
f et a doivent avoir la même longueur. La longueur doit être un nombre pair.
firpm utilise toujours un ordre de filtrage pair pour les configurations à symétrie paire et une bande passante non nulle à la fréquence de Nyquist. En effet, pour les réponses impulsionnelles présentant une symétrie paire et des ordres impairs, la réponse en fréquence à la fréquence de Nyquist est nécessairement 0. Si vous spécifiez un n impair, firpm l'incrémente de 1.

b = firpm(n,f,a,w) utilise les poids du vecteur w pour pondérer l'ajustement dans chaque bande de fréquences. La longueur de w est la moitié de la longueur de f et de a, il y a donc exactement un poids par bande.

b = firpm(n,f,a, 'ftype') et
b = firpm(n,f,a,w, 'ftype') spécifient un type de filtre, où 'ftype' est 'hilbert', pour les filtres à phase linéaire à symétrie impaire (type III et type IV)
Les coefficients de sortie en b obéissent à la relation b(k) = -b(n + 2 - k), k = 1, ..., n + 1. Cette classe de filtres comprend le transformateur Hilbert, qui a une amplitude désirée de 1 sur toute la bande.
Par exemple,
h = firpm(30, [0,1 0,9], [1 1], 'hilbert') ;
conçoit un transformateur Hilbert FIR approximatif de longueur 31.
'différenciateur', pour les filtres de type III et de type IV, en utilisant une technique de pondération spéciale.

```
f = [0 0.3 0.4 0.6 0.7 1];
a = [0 0.0 1.0 1.0 0.0 0];
b = firpm(17,f,a);
[h,w] = freqz(b,1,512);
plot(f,a,w/pi,abs(h))
legend('Ideal','firpm Design')
xlabel 'Radian Frequency (\omega/\pi)', ylabel 'Magnitude'
```

II Travail à réaliser sur Matlab

II.1 Synthèse par la méthode de la fenêtre

1. Ecrire un script Matlab qui permet d'étudier les réponses temporelles et fréquentielles de plusieurs fenêtres : rectangulaires, Bartlett, Hamming, Hanning, Blackman… en utilisant les fonctions matlab : rectwin(N), hanning(N) et blackman(N)…
Comparer la taille des lobes secondaires et la largeur du lobe principal des différentes fenêtres.

2. Nous voulons concevoir, en utilisant la méthode de la fenêtre, un filtre numérique RIF de type passe-bas qui obéit aux caractéristiques suivantes : bande passante de 0 à 2400 Hz, bande atténuée au-delà de 3 kHz avec une ondulation minimale de 40 db. La fréquence d'échantillonnage est de 8000 Hz.
2.1 Déterminer la valeur du paramètre *fn* à introduire dans la fonction fir1.
2.2 Écrire le script Matlab permettant de faire la synthèse de ce filtre en utilisant une fenêtre rectangulaire et en faisant varier l'ordre du filtre de N=10 jusqu'à N=100.
2.3 Visualiser la réponse impulsionnelle et la réponse fréquentielle en module des filtres RIF obtenus.
2.4 Comparer les réponses fréquentielles obtenues pour les différentes valeurs de N et observer son influence sur la raideur de la pente et sur la position de la fréquence de coupure à 3dB.
2.5 Trouver l'ordre minimal satisfaisant les spécifications.

3. Synthétiser maintenant le filtre à l'aide d'autres types de fenêtres disponibles (triangulaire, Hamming, Hanning, Blackman et Kaiser).
3.1 Visualiser la réponse impulsionnelle et la réponse fréquentielle en module et en phase des filtres RIF obtenus.
3.2 Comparer les réponses fréquentielles obtenues par les différentes fenêtres : amplitude des ondulations en bande passante et en bande affaiblie, valeur de la pente. Commenter.
4. Toujours avec la méthode de la fenêtre, nous voulons concevoir un filtre numérique RIF de type passe-bande qui obéit aux caractéristiques suivantes : bande passante [1000Hz 2400Hz], largeur de la zone de transition $\Delta f = 600\ Hz$. La fréquence d'échantillonnage est de 8000 Hz.
Écrire le script Matlab permettant de synthétiser plusieurs filtres avec des valeurs croissantes de l'ordre N et en utilisant plusieurs fenêtres.

mibousahla@gmail.com

II.2 Synthèse par la méthode de l'échantillonnage fréquentiel

Nous voulons concevoir les mêmes filtres passe-bas et passe-bande par la méthode de l'échantillonnage fréquentiel. Écrire le programme Matlab permettant de faire cette synthèse en suivant les mêmes questions qu'à la partie A. Comparer les résultats obtenus avec la méthode des fenêtres.

II.3 Synthèse par les méthodes d'optimisation

Synthétiser les mêmes filtres passe-bas et passe-bande par les méthodes d'optimisation des moindres carrés (firls) et de Parks-McClellan (firpm).

mibousahla@gmail.com

Bibliographie

[1] Traitement Numérique du Signal, M. Bellanger, Ed Masson, Collection CNET-ENST.

[2] Maurice Charbit, Gérard Blanchet. Eléments de base pour le Traitement Numérique du Signal et de l'Image. Telecom Paris.

[3] Olivier Sentieys. Signaux et Systèmes Discrets. ENSSAT - Université de Rennes 1.

[4] Francis Cottet. Aide-mémoire Traitement du Signal. Dunod, Paris.

[5] J.L. Zarader. Cours de Traitement du Signal, première partie. Ecole Polytechnique Universitaire de Paris

[6] J.L. Zarader. Cours de Traitement du Signal, seconde partie. Ecole Polytechnique Universitaire de Paris

[7] Hugues GARNIER. Conception de filtres numériques. Université de lorraine, Polytech Nancy.

[8] Filtrage Numérique, Thierry PAQUET, UFR des Sciences.