TRAITEMENT NUMÉRIQUE DU SIGNAL AVEC MATLAB

Traitement du Signal - TD 5

Université de Lorraine - ENSEM ISN 2A

FRANÇA DE SALES Déric Augusto

32219632

deric-augusto.franca-de-sales6@etu.univ-lorraine.fr

FERREIRA MARTINS Michelle

32219634

michelle.ferreira-martins5@etu.univ-lorraine.fr

06 janvier 2023

1 Introduction

Le filtrage des signaux est utilisé dans plusieurs domaines, comme dans l'analyse des signaux biomédicaux bruyants au travail avec les applications d'instruments industriels ou même des signaux sonores. Dans l'analyse des signaux oscillatoires bruyants, les filtres sont fondamentaux, car ils permettent d'éliminer certaines fréquences d'un spectre, générant un signal plus propre ou même séparant ces fréquences. Il est donc nécessaire d'utiliser la transformée de Fourier pour faire passer l'analyse des signaux du domaine temporel au domaine fréquentiel. Il est possible d'effectuer des opérations plus simples dans l'un des domaines, puis d'utiliser la transformée de Fourier pour revenir dans l'autre domaine.

En ce qui concerne les signaux sonores, par exemple, il est possible d'appliquer cette technique décrite pour l'élimination des bruits de fond générés par un enregistrement, l'application de filtres pour générer des effets sur les signaux sonores, ou encore la séparation des pistes audio contenues dans un seul fichier audio. Il est possible, par exemple, d'identifier les fréquences dominantes de la voix et des instruments dans une chanson, puis de les séparer, en générant des pistes audio distinctes contenant dans chacune d'elles, chacun des éléments audio.

Ainsi, pour analyser un signal, il faut d'abord l'échantillonner. Cet échantillonnage traitera de la transformation d'un signal continu en un signal discret, comme par exemple lors de l'analyse d'un signal sonore réel, où une séquence d'échantillons mesurés dans un ensemble fini d'instants de temps est collectée. Et après son échantillonnage, est fortement utilisé l'algorithme de la transformée de Fourier rapide (FFT), pour analyser le signal dans le domaine des fréquences, qui calcule la transformée de Fourier à une complexité réduite, exigeant moins de la capacité du ordinateur.

Dans ce rapport, nous chercherons à atteindre les objectifs décrits ci-dessous, en suivant le plan proposé par le matériel fourni. Les graphiques et autres résultats ont été générés à l'aide du MATLAB v.2020a. Les algorithmes écrits et les opérations effectuées sont joints en annexe.

2 Objectifs

— Étudier l'échantillonnage d'un signal ;
— effectuer l'analyse de son spectre par l'algorithme FFT ;
— analyser et filtrer un signal numérique par différentes méthodes.

3 Échantillonnage d'un signal porte

3.1 Détermination de la fréquence d'échantillonnage Fe en appliquant Shannon

Initialement, un signal de porte d'amplitude 1 et de largeur d'une seconde (allant de -0.5 à 0.5 secondes) a été défini. Le signal tracé dans matlab est défini dans la figure 1.

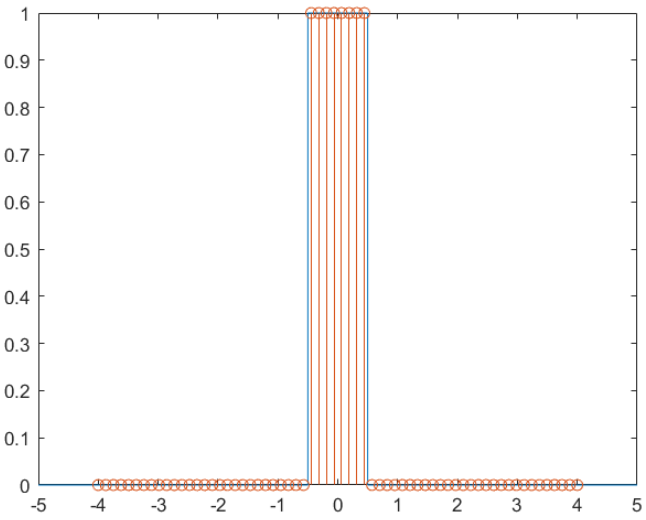

FIGURE 1 – Signal porte échantillonée.

Dans le cas réel, la fonction démontrée ci-dessus présenterait une infinité de fréquences génératrices, mais pour l'application en question, nous allons considérer un nombre limité de fréquences, nous allons définir la fréquence d'échantillonnage comme $Fe = 8Hz$ et la résolution fréquentielle comme $Rf = 0.125$. Ainsi, il est possible définir la période d'échantillonnage du signal ($Te = \frac{1}{Fe}$) et le nombre de points pour tracer la fonction porte $N = \frac{Fe}{Re}$.

3.2 Élaboration d'un signal numérique x(n) obtenu par échantillonnage de x(t)

Après la mise en pot du signal rectangulaire, il a été possible de calculer le spectre du signal en utilisant la commande matlab *fftshift(abs(fft(y)))*, où y est le signal dans le domaine temporel. La réponse à cette commande sera le vecteur courant du signal échantillonné dans le domaine fréquentiel. Le résultat du tracé peut être vérifié dans la figure 2.

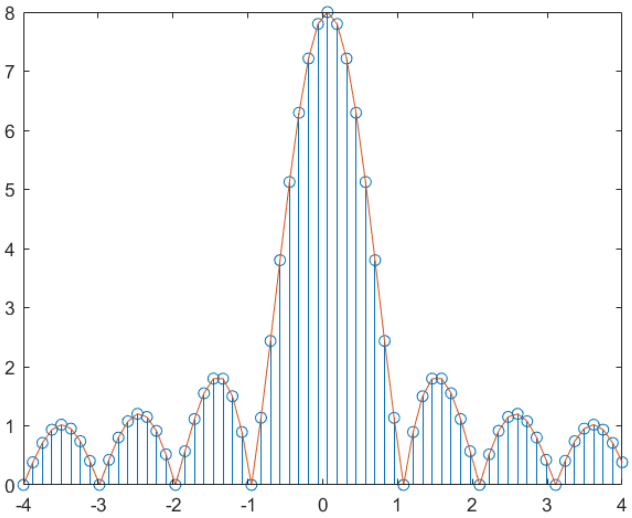

FIGURE 2 – Spectre du signal porte générée.

Après cette étape, un signal créé par la somme de deux sinusoïdes avec une addition de bruit était encore généré. Le signal généré par la relation 1 peut être vérifié par la figure 3. Il est intéressant de noter que pour ce signal, une fréquence d'échantillonnage de $7000Hz$ et une résolution de fréquence de 35 ont été fixées.

$$y = 4 \cdot sin(2 \cdot \pi \cdot 1500 \cdot x) + 6 \cdot sin(2 \cdot \pi \cdot 3500 \cdot x) \tag{1}$$

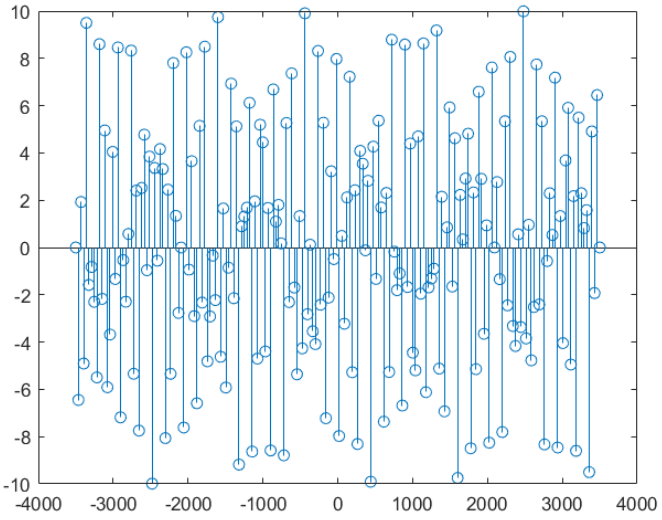

FIGURE 3 – Signal y (1) dans le domaine temporel.

Après avoir défini le signal, on a procédé de la même manière que pour le signal porte et on a obtenu le spectre du signal y présenté à la figure 4. Il est intéressant de noter comment les fréquences génératrices du signal y ont été mises en évidence dans son spectre.

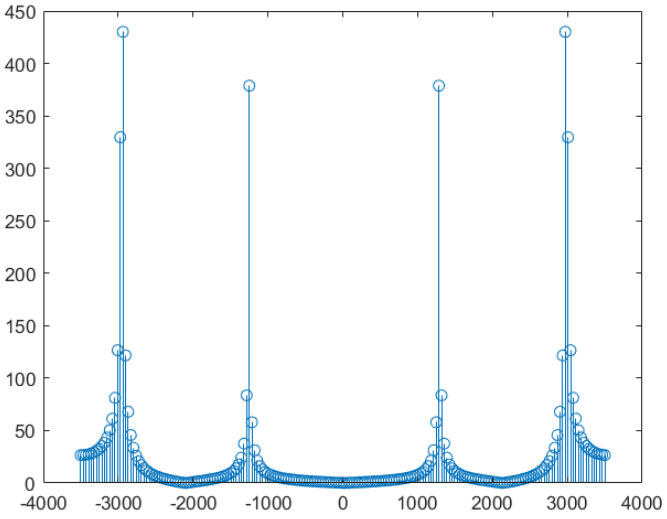

FIGURE 4 – Spectre du signal y.

4 Traitement numérique d'un signal par filtrage IIR et FIR

Tout d'abord, le fichier "mesure.dat" a été chargé et à partir des données obtenues, la figure 5 a été tracée. A partir du signal chargé, il a été possible de définir la période d'échantillonnage, sa fréquence d'échantillonnage et sa durée, comme écrit dans l'algorithme ci-joint. Une fois les données chargées, le spectre du signal mis à disposition par la figure 6 a été généré. Grâce au spectre, il est alors possible de définir que le signal est formé par un bruit et deux fréquences fondamentales de 500 et 1200Hz.

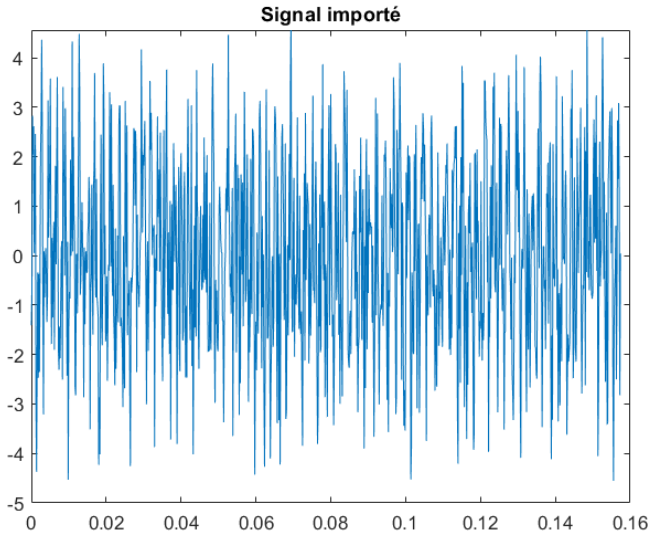

FIGURE 5 – Signal importé.

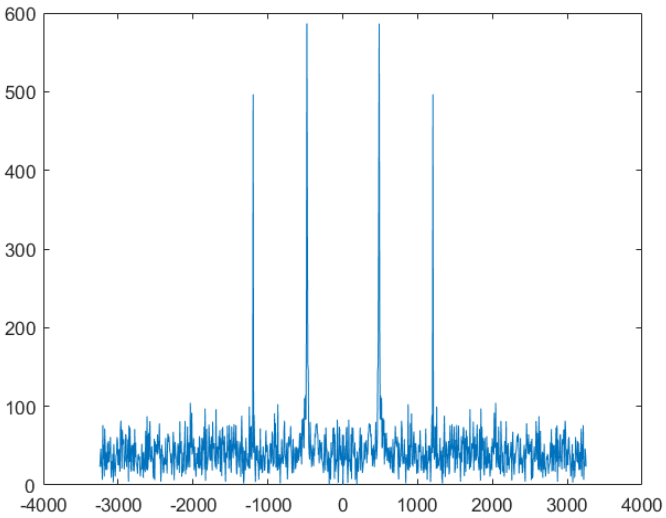

FIGURE 6 – Spectre du signal importé.

4.1 Filtre *IIR* (Infinite Impulsion Response filter)

Les filtres *IIR* sont des filtres discrets caractérisés par une réponse impulsionnelle qui ne devient pas exactement nulle après un certain point, mais continue indéfiniment. C'est le contraire de ce qui se passe avec les filtres *FIR*, qui seront abordés ensuite, où la réponse impulsionnelle devient exactement nulle après un temps fini. Ainsi, pour définir un filtre *IIR* dans matlab, on définit d'abord son ordre et sa gamme de fréquence. Nous définissons un ordre de 50 et une gamme de fréquences de 400 à 1300Hz.

Après avoir appliqué le filtre, l'outil matlab "fvtool" a été utilisé pour générer en même temps plusieurs graphiques comme celui de Bode, e tracé pôle-zéro, le délai de groupe et autres.

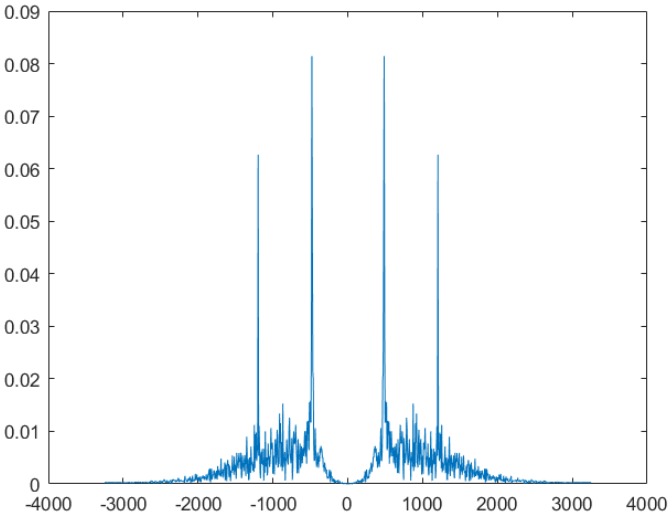

FIGURE 7 – FFT du signal importée après la filtrage *IIR*.

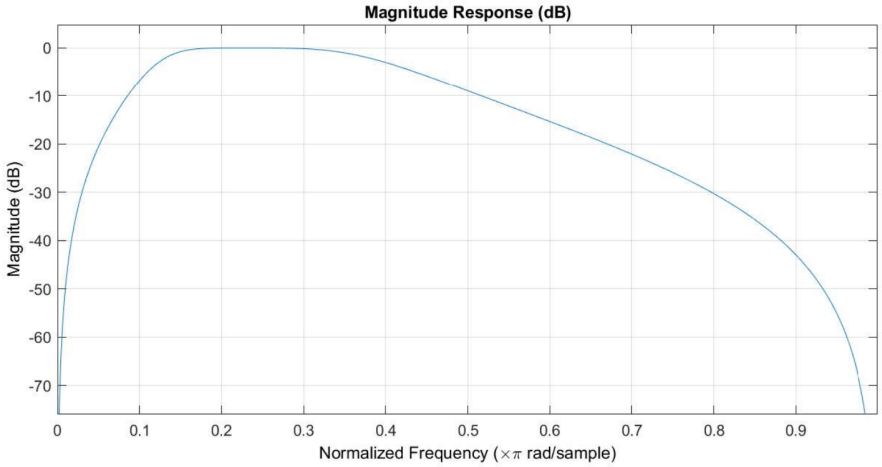

FIGURE 8 – Réponse de magnitude du signal après l'application du filtre *IIR*.

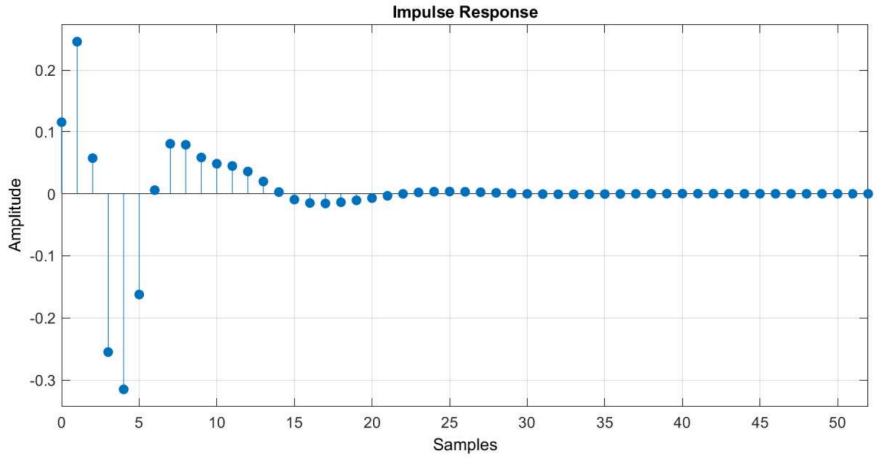

FIGURE 9 – Réponse impulsionnel du signal après l'application du filtre *IIR*.

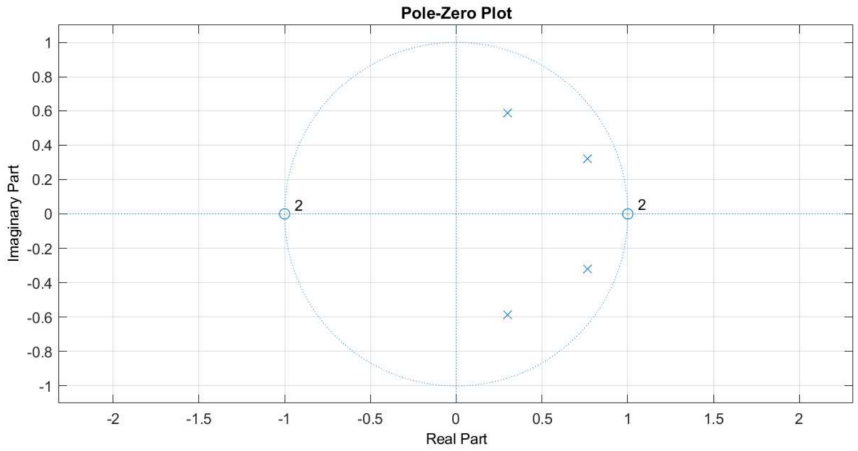

FIGURE 10 – Tracé pôle-zéro du signal après l'application du filtre *IIR*.

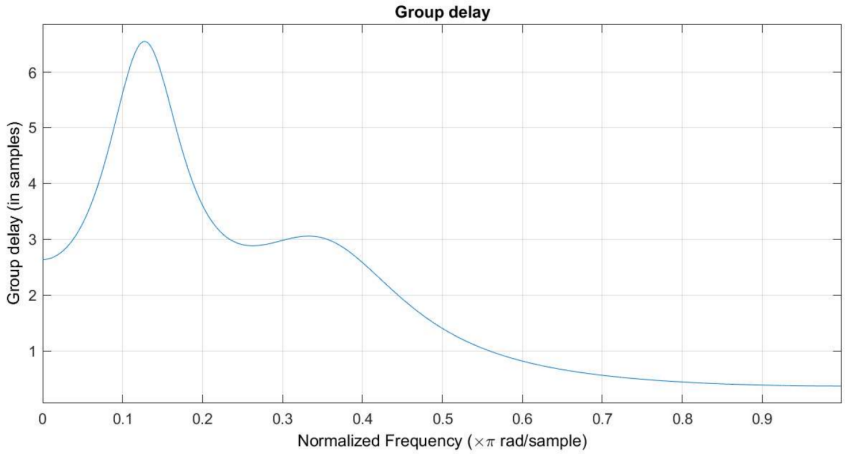

FIGURE 11 – FFT du signal importée après la filtrage *IIR*.

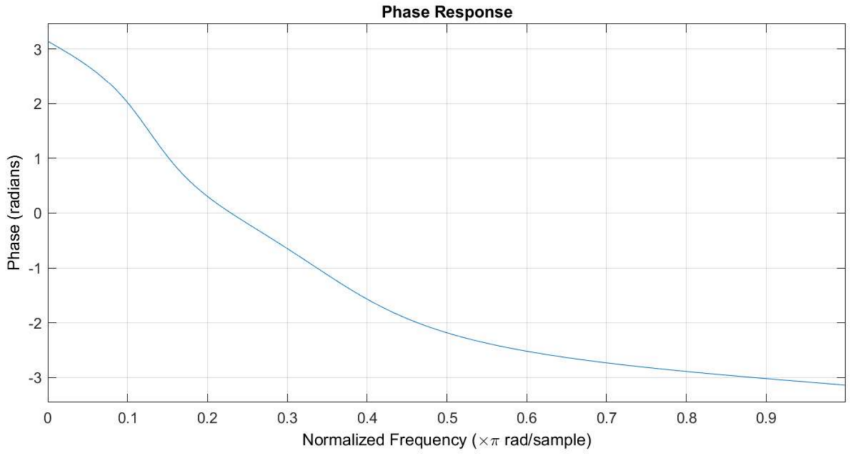

FIGURE 12 – Réponse de phase du signal après l'application du filtre *IIR*.

En analysant le diagramme de Bode de la figure 8, on constate que le gain est de 0 dB pendant la gamme de fréquences choisie, et qu'il y a une atténuation pour les autres valeurs. Le choix de l'ordre du filtre et la fréquence d'échantillonnage influencent la réponse impulsionnelle du filtre. En analysant le graphique de la figure 9, on constate qu'une réponse impulsionnelle d'environ 25 points a été générée. Il est important de se rappeler que les filtres *IIR* ne sont pas toujours stables. Cette caractéristique peut être étudiée par le biais du plan complexe. Pour être un filtre stable, les pôles doivent se trouver dans le cercle unitaire.

4.2 Filtre *FIR* (finite impulse response filter)

De même que ce qui a été fait pour le filtre *IIR*, cela a été fait pour le filtre *FIR*, générant les figures 13, 14, 15, 16, 17 et 18.

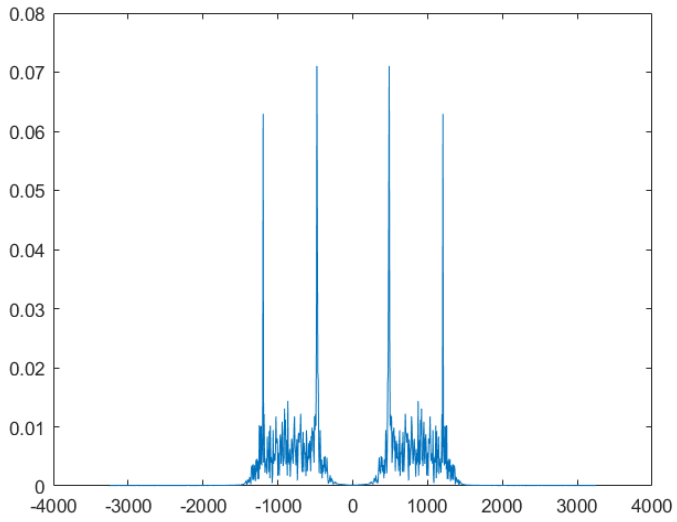

FIGURE 13 – FFT du signal importée après la filtrage *FIR*.

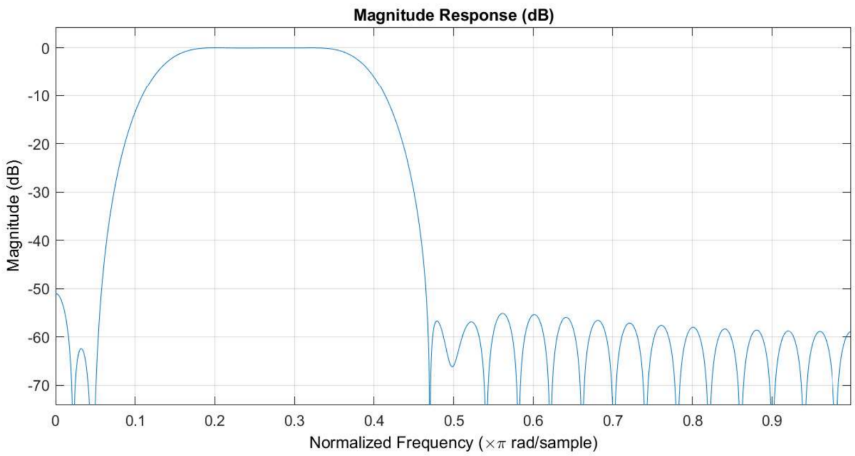

FIGURE 14 – Réponse de magnitude du signal après l'application du filtre *FIR*.

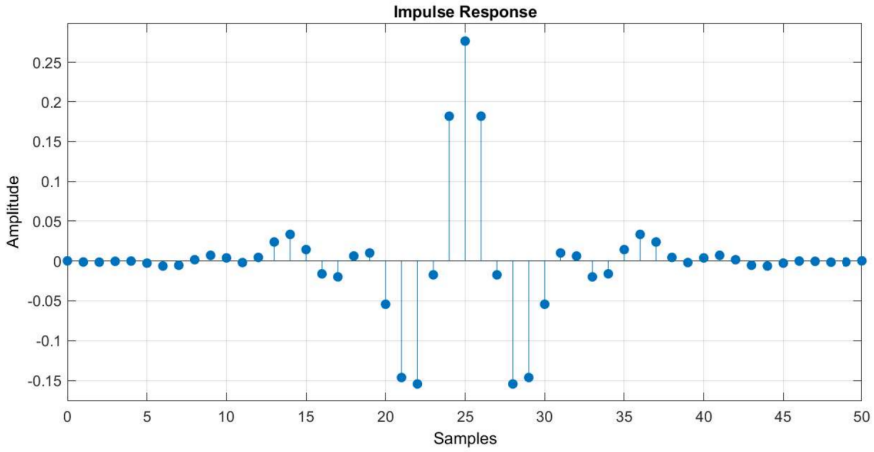

FIGURE 15 – Réponse impulsionnelle du signal après l'application du filtre *FIR*.

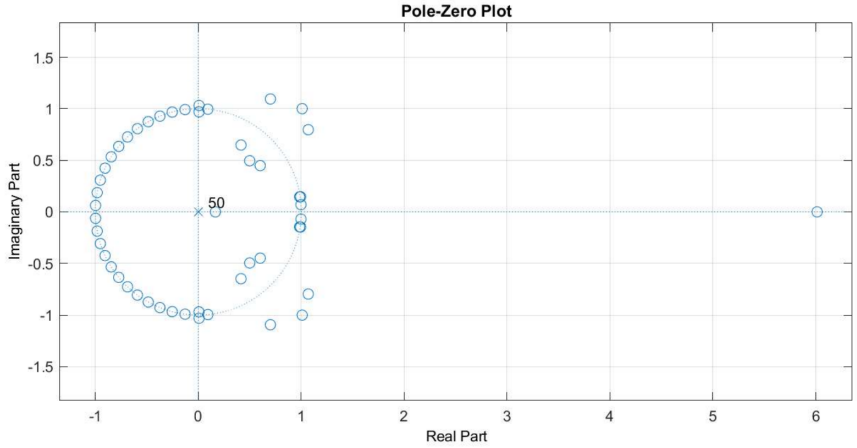

FIGURE 16 – Tracé pôle-zéro du signal après l'application du filtre *FIR*.

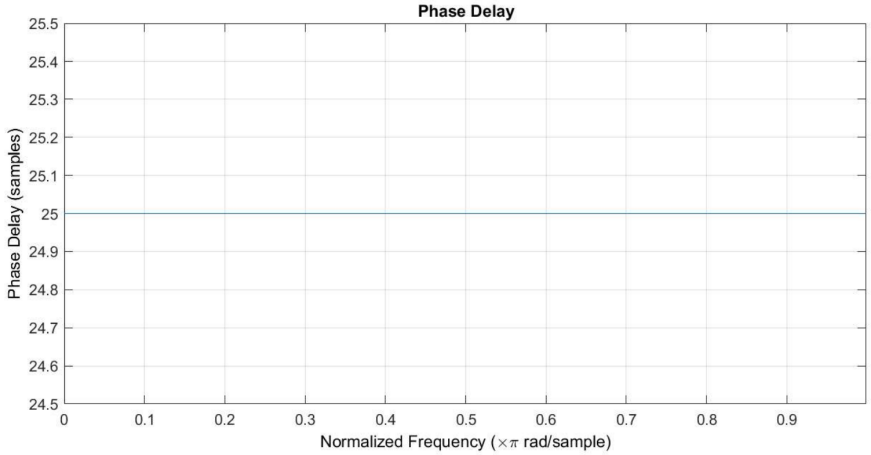

FIGURE 17 – Tracé pôle-zéro du signal après l'application du filtre *FIR*.

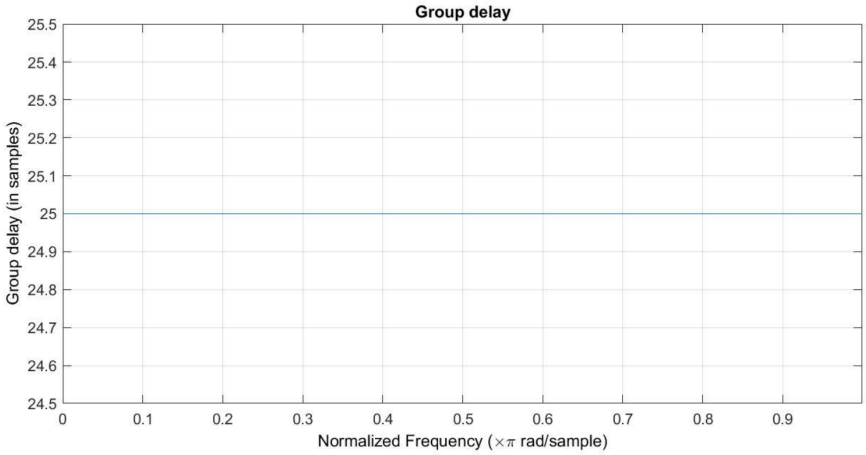

FIGURE 18 – du signal après l'application du filtre *FIR*.

Contrairement au filtre *IIR*, la réponse impulsionnelle du filtre *FIR* nécessite que plusieurs échantillonnages soient effectués afin d'obtenir la stabilité. Comme il n'a pas de pôles, il n'est pas possible de fixer une valeur maximale. D'autre part, les zéros sont réglés sur les valeurs souhaitées. En analysant le plan complexe, on observe qu'il existe des zéros sur le cercle unitaire et à son extérieur. Les premières représentent un gain nul, tandis que les autres représentent les fréquences à maintenir. Le filtrage peut être amélioré en augmentant l'ordre du filtre. Lorsque cela se produit, le filtre *FIR* ressemble de plus en plus au filtre *IIR*. En contrepartie, la réponse impulsionnelle, le gain et la position des zéros sont impactés.

Annexe

Signal Processing Toolbox

14/12/2022

summary

Déric Augusto França de Sales

Michelle Ferreira Martins

```
clc; close all; clear all;
```

A) Echantillonnage d'un signal port

1. Détermination de la fréquence d'échantillonnage

```
Rf = 0.125; % resolution frequentielle
Fe = 8; % frequence d'echantillonnage
```

2. Elaborer le signal numérique x(n) obtenu par échantillonnage de x(t).

```
syms a
signal_continu = rectangularPulse(a), [-1 1];
fplot(signal_continu) % signal porte continu d'amplitude 1 de largeur 1 seconde
hold on
x = linspace(-Fe/2, Fe/2, Fe/Rf);
y = [zeros(1, Fe/(2*Rf) - Fe/2), ones(1, Fe), zeros(1, Fe/(2*Rf) - Fe/2)];
stem(x, y) % signal porte discret d'amplitude 1 de largeur 1 seconde
%title('Signal porte échantillonée')
```

3. Calculer le spectre de x(n) par et tracer son spectre.

```
figure
z = fftshift(abs(fft(y)));
stem(x, z) % transformee de fourier discrete
hold on
plot(x, z) % transformee de fourier continue
%title('Spectre du signal porte')
```

Signal extra

```
Fe_new = 7000;
Rf_new = 35;
```

```
x_new = linspace(-Fe_new/2, Fe_new/2, Fe_new/Rf_new);
y = 4*sin(2*pi*1500*x_new) + 6*sin(2*pi*3500*x_new);
figure
stem(x_new, y);
figure
z = fftshift(abs(fft(y)));
stem(x_new, z)
```

B) Traitement numérique d'un signal par filtrage IIR et FIR

1. Importe du fichier mesure.dat

```
load ('mesure.dat');
t = mesure(1,:); % vecteur temps
x = mesure(2,:); % signal échantillonne
taille = length(x); % nombre de points
figure
plot(t,x) % Signal importe
%title('Signal importé')
```

2. Analyse du signal

```
Te = mesure(1,2) - mesure(1,1); % periode d'echantillonnage
Fe = 1/Te; % frequence d'echantillonnage
duree = mesure(1,1024); % duree du signal

f = -Fe/2 : 1/duree : Fe/2;
y = fftshift(fft(x));
figure
plot(f,abs(y))
%title('Spectre du Signal') % FFT
```

3. Filtrage du signal

Technique RII

```
order_RII = 2; % filter's order
pass_band_RII = [400 1300]*2/Fe; % frequency range

[b, a] = butter(order_RII, pass_band_RII);

fvtool(b,a); % generates bode graphs (magnitude and phase),
%pole and zero diagrams and others
fRII = filter(b, a, x);
fft_RII = fftshift(abs(fft(fRII))/Fe);
figure
plot(f, fft_RII)
%title('FFT du signal après le ftilrage RII')
```

Technique RIF

```
order_RIF = 50; % filter's order
pass_band_RIF = [400 1300]*2/Fe; % frequency range

RIF = fir1(order_RIF, pass_band_RIF);
fvtool(RIF, 1); % generates bode graphs (magnitude and phase),
%pole and zero diagrams and others
fRIF = filter(RIF, 1, x);
fft_RIF = fftshift(abs(fft(fRIF))/Fe);
figure
plot(f, fft_RIF)
%title('FFT du signal après le filtrage RIF')
```