JAMBON Antoine
REVEILLER Baptiste

21/11/2018

Traitement de Signal

Compte rendu TD n°5

Objectifs :

- Savoir échantillonner un signal et élaborer son spectre par FFT
- Savoir analyser et filtrer un signal numérique à partir des méthodes classiques

A) Echantillonnage d'un signal porte.

En toute rigueur, on ne peut pas appliquer Shannon : **Fe>2*fmax** car ici **fmax** est infini pour la fonction porte. On va donc considérer que f est bornée dans ce problème.

Il faut donc trouver la transformée de Fourier du signal porte.

Tout d'abord nous traçons la fonction porte sur ***Matlab*** avec le code suivant :

```
%% Enchatillonage d'un signal porte

Fe = 20; % fréquence d'échantillonage
Rf = 0.125; % résolution fréquentielle

Te = 1/Fe; % période d'échantillonage
N = Fe/Rf; % nombre de points

TempInterv = 1; % en seconde
N1= TempInterv/Te;
x = linspace(-Rf*(N/2),Rf*(N/2),N);

pN = [x ; zeros(1,N)]';
pN((N/2)-(N1/2)+1:(N/2)+(N1/2),2)=1;

figure(1)
plot(pN(:,1),pN(:,2),'r')
axis([-(N/2)*Rf-0.1 (N/2)*Rf+0.1 0 1.1]);
```

Figure 1 - Code Matlab pour construire et tracer la fonction porte

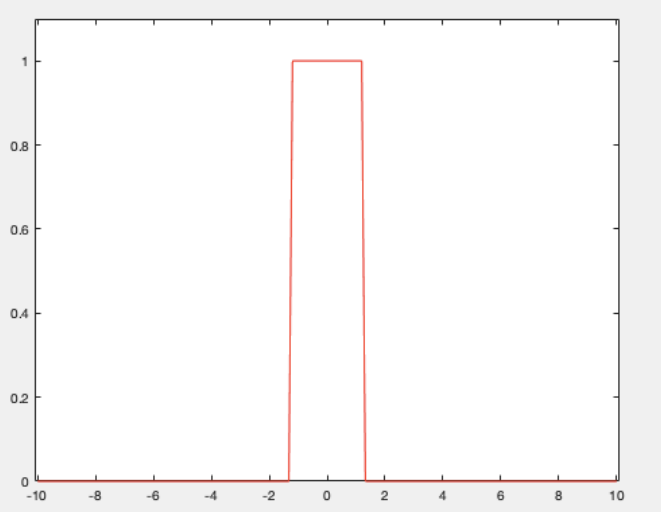

Figure 2 - Représentation temporelle de la fonction porte construite précédemment

On donne *Fe = 8Hz, Rf = 0.125* (fréquence d'échantillonnage et résolution fréquentielle), ainsi on peut trouver notre période d'échantillonnage Te = 1/Fe et notre nombre de points pour tracer la fonction porte, *N = Fe/Re.*

On définie *x* le vecteur des abscisses qui dépend de notre nombre de points N et notre résolution fréquentielle. Par la suite on écrit *pN* une matrice correspondant aux *y (ordonnées)* de notre fonction porte, puis nous écrivons l'équation pour avoir une amplitude égale à 1.

$$P(f)=\int_{-\infty}^{+\infty} p(t)e^{-j2\pi ft}dt=\int_{-\frac{1}{2}}^{\frac{1}{2}} e^{-j2\pi ft}\,dt=\left[-\frac{1}{j2\pi ft}e^{-j2\pi ft}\right]_{-\frac{1}{2}}^{\frac{1}{2}}=\mathrm{sinc}(\pi f)$$

Et on sait que : $\text{sinc}(\pi f) = 0 \Rightarrow \pi f = k\pi \Rightarrow f = k$

On obtient ainsi la réponse fréquentielle du sinus cardinal :

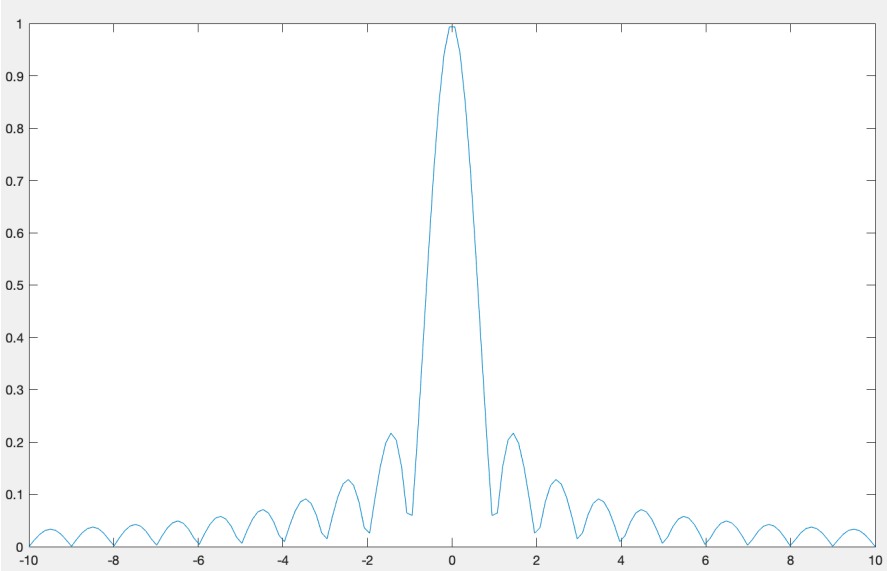

Figure 3 - Représentation fréquentielle de la fonction sinus cardinal

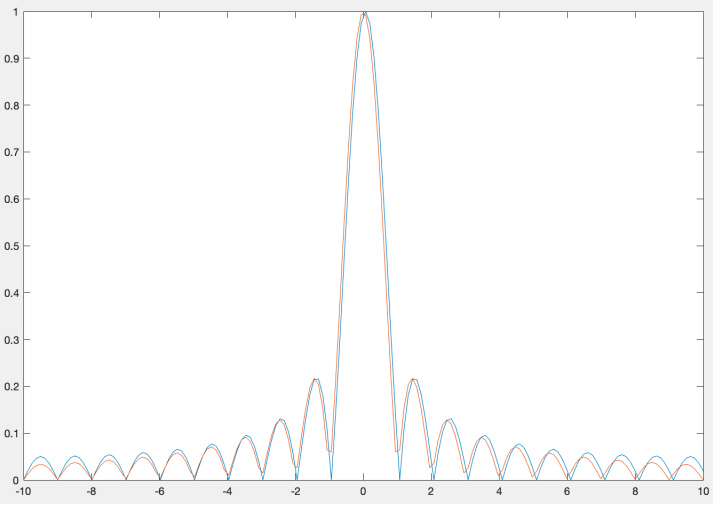

Figure 5 - Représentation fréquentielle de la FFT superposée avec le sinus cardinal précédant

```
Yfft=fftshift(abs(fft(pN(:,2)))*Te);
Ycard=abs(sinc(x));

figure(2)
plot(x,Yfft,x,Ycard)
```

Figure 4 - Code Matlab pour effectuer la FFT et le tracer ainsi que le sinus cardinal théoriquement obtenu

La fonction ***fftshift()*** est la fonction *Fast Fourier Transformation* qui effectue la transformé de Fourier en centrant l'intervalle des fréquences sur la fréquence 0 Hz.
On applique la fonction ***abs()*** pour avoir des ordonnées réelles lors de la première ***fft()***.

Pour avoir l'amplitude des fréquences souhaitée, il faut multiplier les ordonnées par Te après la transformée de Fourier pour obtenir le sinus cardinal calculé précédemment.

La différence entre le sinus cardinal théorique (en rouge ici) et la FFT du signal porte (en bleue) est dû au nombre de points de l'échantillonnage, en effet plus le nombre de points est grand et la fréquence d'échantillonnage petite plus les deux courbes tendent à se superposer.

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

On définit une nouvelle fréquence d'échantillonnage *Fe2 = 1000 Hz*, une résolution fréquentielle *Rf2 = 0.1* et un nouveau nombre de points *N2 = Fe2 / Rf2*. Ainsi on détermine une base temporelle ***baset*** et une base fréquentielle ***basef*** pour pouvoir tracer ***Xt*** en temporelle puis son spectre en fréquentielle. Voici ce qu'on obtient :

```
%% Question supplémentaire
Fe2 = 1000;
Rf2 = 0.1;
N2 = Fe2/Rf2;

baset = 0:1/Fe2:(N2-1)*1/Fe2;
basef = -Fe2/2:Rf2:(Fe2/2)-Rf2;

Xt = 3*cos(2*pi*100*baset)+6*sin(2*pi*400*baset);

figure(3)
plot(basef,fftshift(abs(fft(Xt))*1/N2));
```

Figure 6 - Code Matlab avec la génération d'un signal connu Xt et calcul de son spectre

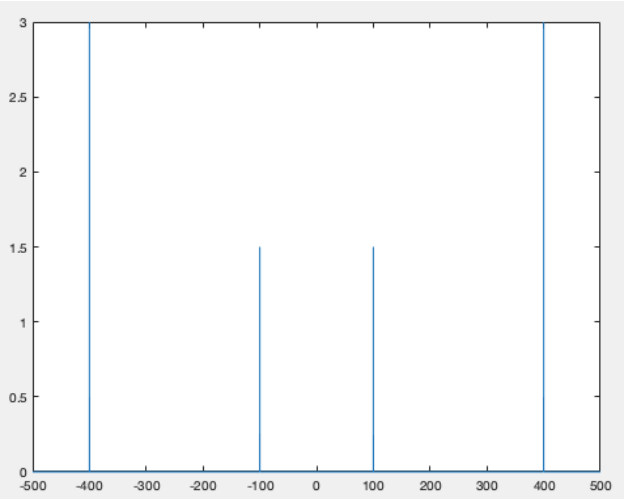

Figure 7 - Représentation du spectre de Xt

On peut voir que l'on a un spectre comprenant 2 signaux, c'est à dire un cosinus et un sinus. Notre cosinus nous donne une fondamentale en 100 Hz d'amplitude *3/2 = 1.5* , puis le sinus une fondamentale en 400 Hz d'amplitude *6/2 = 3.* Pour le tracer du spectre, nous utilisons la même méthode que pour la première partie.

Ensuite nous devons importer le fichier ***mesure.dat.txt*** grâce à la fonction ***load*** sur ***Matlab***.

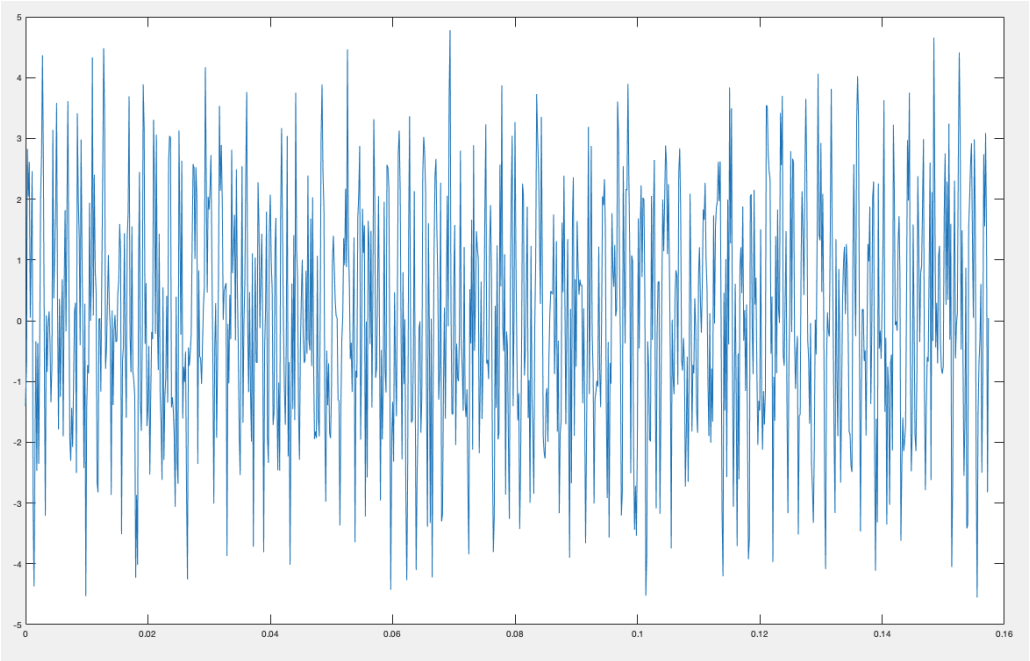

Figure 8 - Représentation temporelle du fichier mesure.dat

Même processus que précédemment, on redéfinie une fréquence d'échantillonnage et une période d'échantillonnage. Puis on trace le spectre du fichier ***mesure.dat.txt***

```
% Analyse du signal

Te3 = baset(2)-baset(1);
Fe3 = 1/Te3;
N3 = length(n);

basef = -Fe3/2:Fe3/N3:Fe3/2-Fe3/N3;

FFT3 = fftshift(abs(fft(n))*Te3);
figure(4)
plot(basef,FFT3)
```

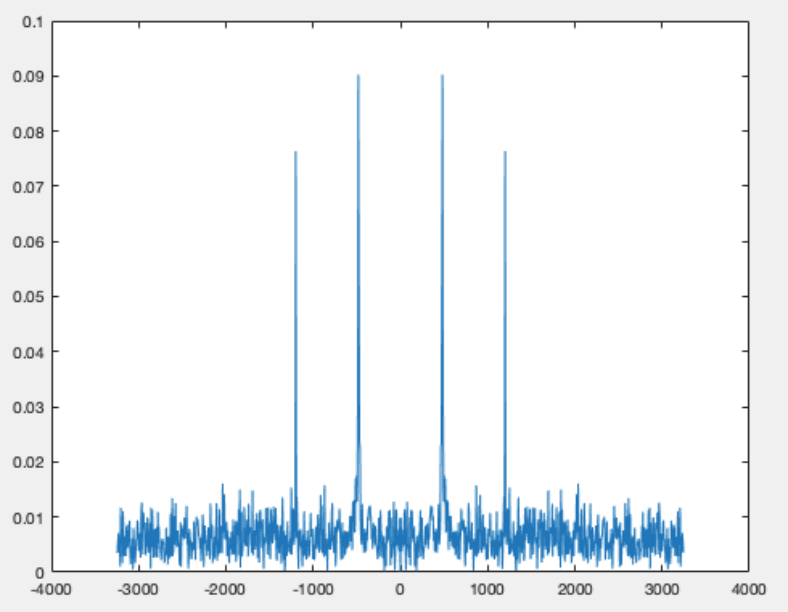

Figure 9 - Représentation spectrale du signal présent dans le fichier mesure.dat

Ce signal présente un phénomène d'aliasing. On observe une fondamentale d'amplitude 0.09 à F01 = 450 Hz pour un premier signal sinusoïdal et une seconde fondamentale d'amplitude 0.077 à la fréquence 1200 Hz.

Filtre FIR (finite impulse response filter) :

On parle le plus souvent de filtre FIR pour des filtres à temps discret. Un filtre numérique FIR est caractérisé par une réponse uniquement basée sur un nombre fini de valeurs du signal d'entrée. Par conséquent, quel que soit le filtre, sa réponse impulsionnelle sera stable et de durée finie, dépendante du nombre de coefficients du filtre.

On donne dans ***Matlab*** l'ordre et la plage des fréquences de filtrage pour définir le filtre FIR de type passe-bande, ici l'ordre est de 50 et l'intervalle de fréquence est de [400Hz 1300Hz].

```
%% FIR

ordre = 50;
wn=[400 1300]*2/FeM;


bFIR = fir1(ordre,wn);
fvtool(bFIR,1)

nfFIR = filter(bFIR,1,n);

YfftFIR = fftshift(abs(fft(nfFIR))*TeM);

figure(7)
plot(basefM,YfftFIR)
```

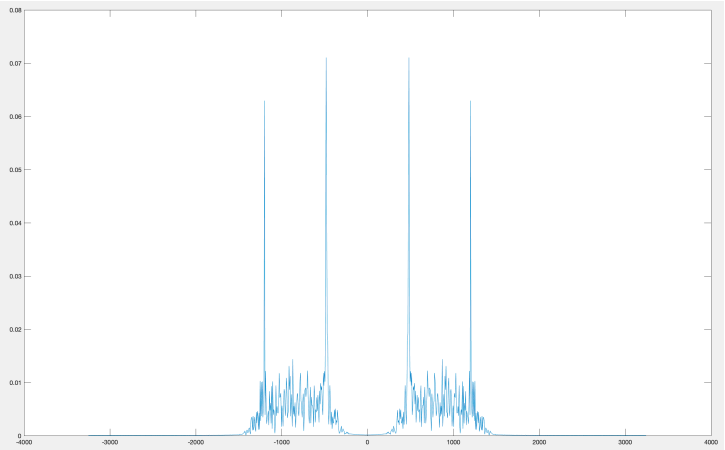

Figure 10 - Représentation spectrale du signal filtré par le FIR

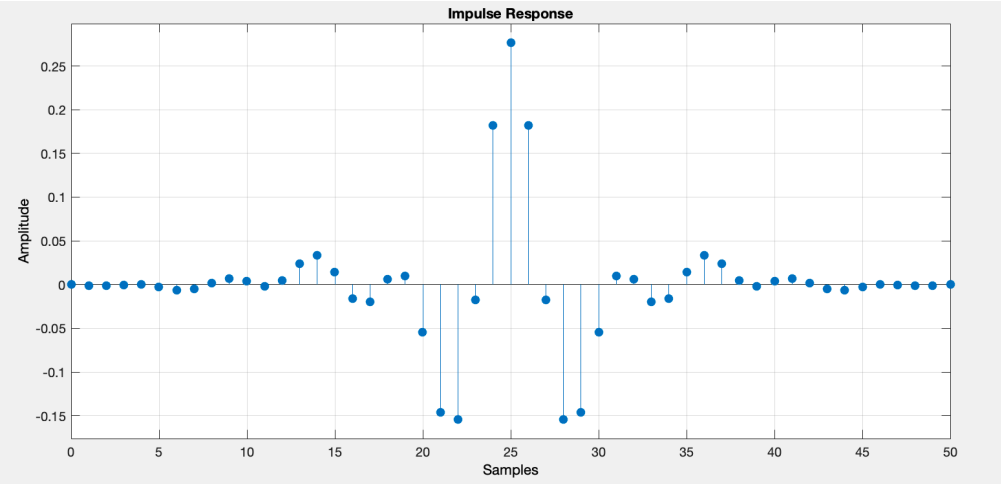

Figure 11 - Représentation de la réponse impulsionnelle

Concernant la réponse impulsionnelle, différente d'un filtre RII, il est nécessaire d'effectuer une centaine d'échantillon pour obtenir une certaine stabilité.

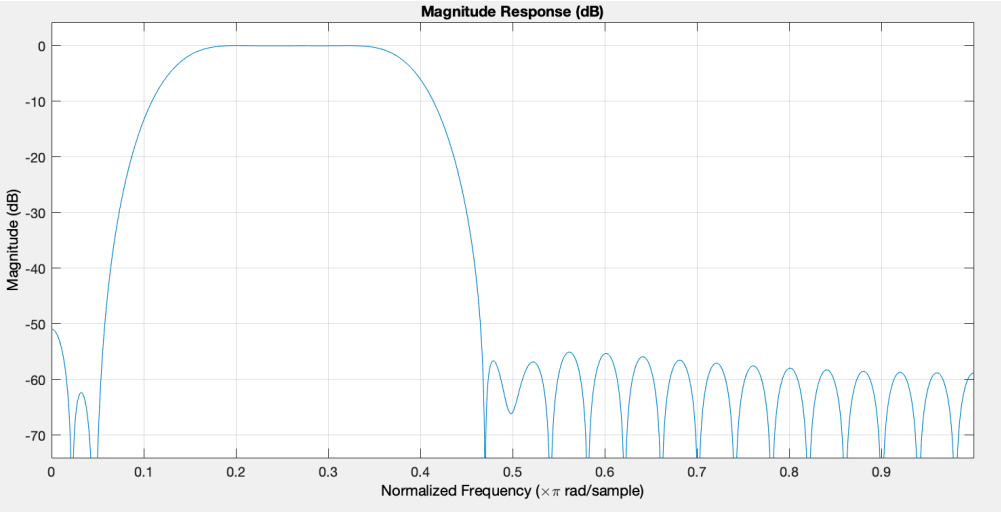

Figure 12 - Représentation en Bode

Comme il n'y a pas de pôles, on ne peut pas fixer de maximum, on peut seulement mettre à zéro aux endroits souhaités ; l'idéal étant de mettre à zéro partout en dehors de la plage de fréquence à garder : ceci explique les rebonds.

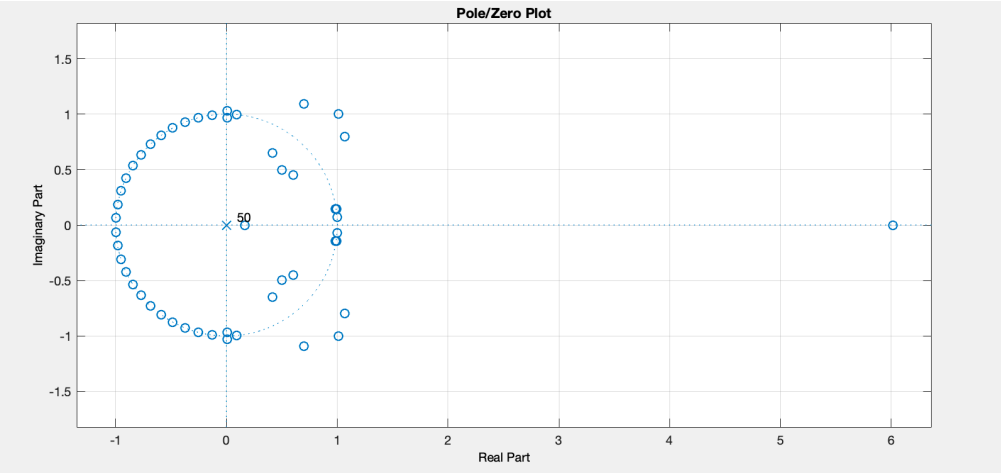

Figure 13 - Représentation des zéros et des pôles dans le plan complexe

Ensuite les zéros qui sont sur le cercle unité signifient que le gain est nul. Les zéros en dehors correspondent aux fréquences à garder. Plus on augmente l'ordre du filtre, plus le phénomène ressemble à celui d'un filtre RII : une légère amélioration du filtrage mais il y a un impact sur la réponse impulsionnelle, le gain et le placement des zéros.

Filtre IIR (Infinite Impulsion Response filter) :

Un filtre à réponse impulsionnelle infinie ou filtre IIR est un type de filtre électronique caractérisé par une réponse fondée sur les valeurs du signal d'entrée ainsi que les valeurs antérieures de cette même réponse.

Dans ***Matlab***, on donne l'ordre du filtre (ici un filtre IIR de type Butterworth) ainsi que l'intervalle de fréquence pour réaliser un filtre passe-bande. Ici, on prend un filtre IIR d'ordre 2 sur l'intervalle [400Hz 1300Hz].

```
%% IIR

ordre = 2;
wn=[400 1300]*2/FeM;

[b,a] = butter(ordre,wn);

fvtool(b,a)
nfIIR = filter(b,a,n);

YfftIIR = fftshift(abs(fft(nfIIR))*TeM);

figure(7)
plot(basefM,YfftIIR)
```

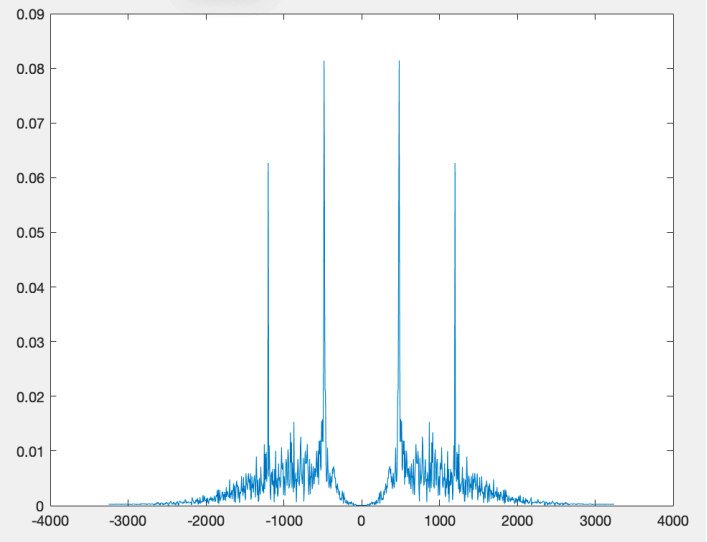

Figure 14 - Représentation spectrale du signal filtré par le IIR

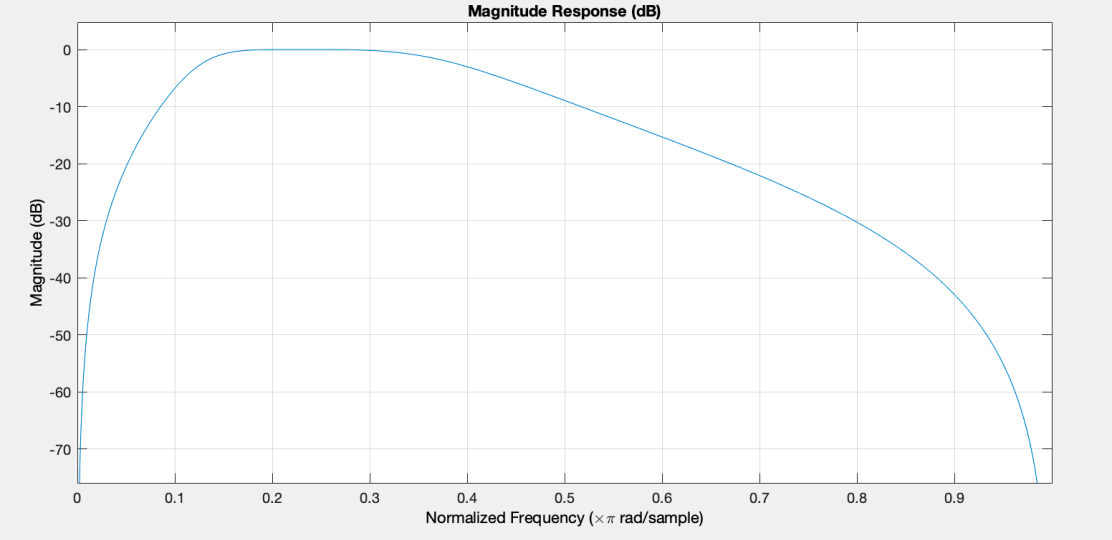

Figure 15 - Représentation en Bode

On observe grâce à la fonction ***fvtool()***, la représentation de Bode de notre filtre de Butterworth IIR. On peut observer qu'il y a une plage où le filtre à un gain de 0 dB (sur l'intervalle choisi précédemment), et une atténuation pour toutes les autres fréquences.

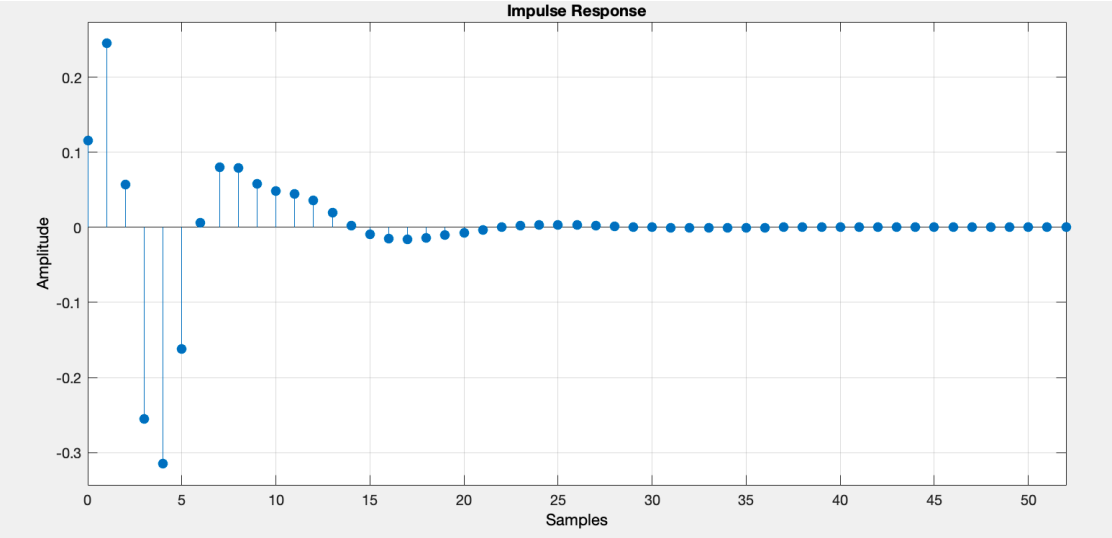

Figure 16 - Représentation en Bode

En construisant notre filtre IIR, on a choisi un ordre et une fréquence d'échantillonnage qui influent sur la réponse impulsionnelle du filtre. Ici, on voit que la réponse impulsionnelle est d'environ 25 points, c'est-à-dire qu'au-delà de 25 points notre filtrage est stable avec ce filtre.

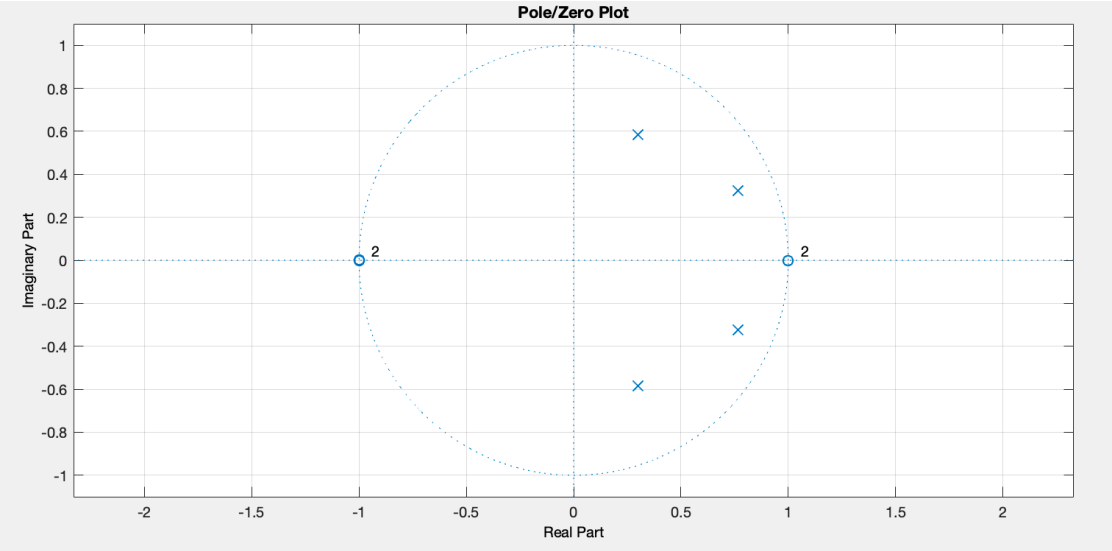

Figure 17 - Représentation des zéros et des, pôles dans le plan complexe

Les filtres RII ne sont pas forcément stables, la stabilité dépend de la position des pôles dans le plan complexe. Pour cela, les pôles doivent être situés dans le cercle unité et non sur le cercle unité.