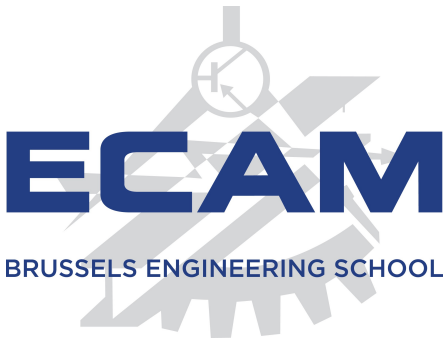


ECAM Brussels Engineering School

Signal Processing

CF5L - FIR Filter Design

Student :
JANSSENS Victor [18322]

Professor :
VAN CAUWENBERGHE Sébastien

Remis le 15 octobre 2023

Table des matières

1 Description Filtre FIR **2**
1.1 Points clés : 2

2 Creation du filtre FIR **3**
2.1 Analyse fichier Audio 3
2.2 Paramètres du filtre FIR 4

3 Conclusion **5**

A Python code **6**
A.1 Ajouter librairies 6
A.2 Ouvrir fichier audio et extraire infos 6
A.3 Transformation de Fourier sample d'origine 6
A.4 Filtre FIR 7
A.5 Transformation de Fourier signal filtré 7
A.6 Réponce fréquentiel du filtre 7

1 Description Filtre FIR

Un filtre FIR, ou filtre à réponse impulsionnelle finie (Finite Impulse Response), est un type de filtre numérique utilisé en traitement du signal pour effectuer des opérations de filtrage sur des signaux numériques. La caractéristique principale d'un filtre FIR est qu'il a une réponse impulsionnelle de durée finie, ce qui signifie que sa réponse à une impulsion unitaire (une impulsion de Dirac) est de durée finie.

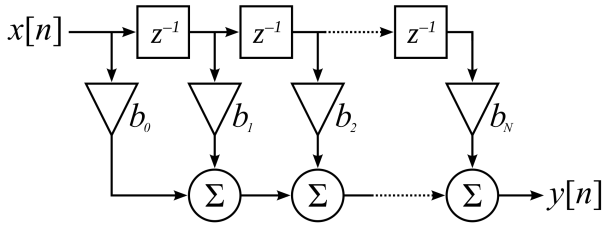


FIGURE 1 – Filtre FIR

1.1 Points clés :

- **Réponse impulsionnelle finie** : La réponse impulsionnelle d'un filtre FIR est constituée d'une séquence finie de coefficients, qui sont les poids appliqués aux échantillons d'entrée pour calculer la sortie. Cette réponse est souvent représentée sous forme de tableau ou de séquence de coefficients.
- **Convolution** : Le fonctionnement d'un filtre FIR est basé sur l'opération de convolution. Pour filtrer un signal d'entrée, le filtre effectue une convolution entre les échantillons du signal d'entrée et les coefficients de la réponse impulsionnelle. Cela donne la sortie filtrée.
- **Réponse en fréquence** : Les coefficients de la réponse impulsionnelle déterminent la réponse en fréquence du filtre FIR. Les fréquences auxquelles le filtre atténue ou amplifie le signal dépendent de ces coefficients. Par conséquent, en ajustant les coefficients, on peut concevoir un filtre qui répond de manière sélective à certaines fréquences.
- **Linéarité et invariant dans le temps** : Les filtres FIR sont linéaires et invariants dans le temps, ce qui signifie que leur comportement ne change pas avec le temps, et ils respectent le principe de superposition, ce qui facilite leur utilisation en traitement du signal.
- **Ordre du filtre** : L'ordre d'un filtre FIR est déterminé par le nombre de coefficients dans sa réponse impulsionnelle. Un filtre d'ordre plus élevé a plus de coefficients et peut fournir une réponse plus précise en fréquence, mais il nécessite également plus de puissance de calcul pour le traitement en temps réel.
- **Conception des filtres FIR** : Il existe différentes méthodes pour concevoir des filtres FIR, telles que la fenêtrage, la méthode des échantillons fréquentiels ou la conception optimale. Chacune de ces méthodes permet de définir les coefficients du filtre en fonction des spécifications de filtrage requises.

2 Creation du filtre FIR

On a un sample de speech en .wav avec un bruit qui a été surimposer sur le sample. On va designer un filtre qui va enlever les fréquences néfastes et récupérer le sample d'origine.

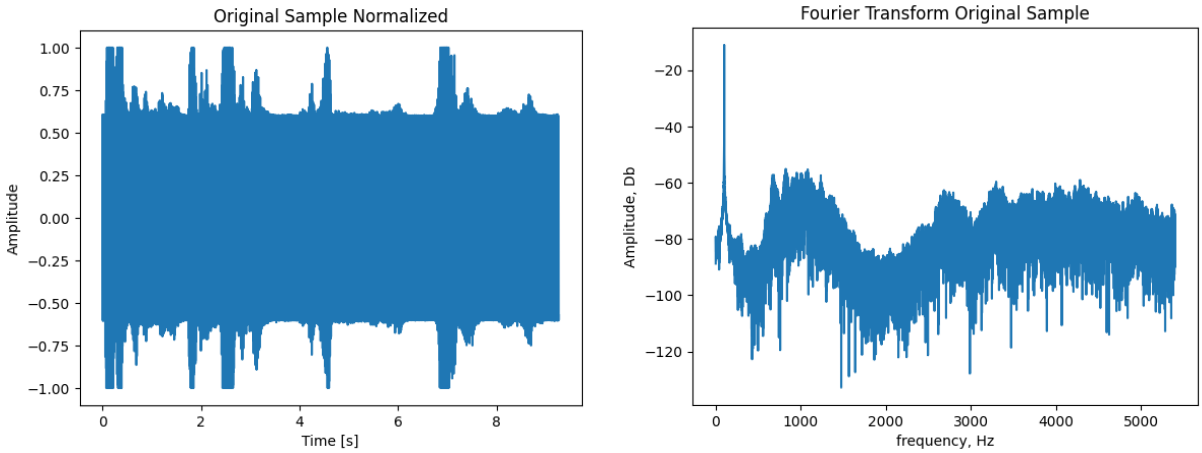
2.1 Analyse fichier Audio

Dans un premier temps on va ouvrir et analyser les fichier audio , Code A.2.

1. **Nombre de channel** : Il y a deux channels dans le fichier audio .wav, donc on va les analyser séparément car les outils de filtrage marche que sur les liste de nombres.
2. **Le type de données** : Les données sont formatter en int16, donc l'amplitude max sera de $\frac{2^{16}}{2} = 32767$ ce qui va nous permettre de tout normaliser par rapport a l'amplitude max du signal audio 2a.

Puis on va appliquer la transformation de Fourier discrète unidimensionnelle sur tout le fichier audio pour les mettre sous forme fréquentielle, Code A.3.

1. **Normalisation par Amplitude de Crête** : On normalise le signal *fft_spectrum* en fonction de l'amplitude max passible du signal.
2. **Normalisation par la taille de l'échantillon** : On va s'assurer que l'amplitude des composantes du spectre est indépendante de la longueur du signal d'entrée. Elle permet de rendre les valeurs du spectre comparables entre différents signaux de longueurs variables.



(a) Original (b) Fast Fourier Transform

FIGURE 2 – Audio Sample

2.2 Paramètres du filtre FIR

Pour créer le filtre FIR il va falloir déterminer les fréquences utiles des fréquences perturbatrices. Dans la téléphonie la bande passante utilisée est entre 300 et 3000 Hz.

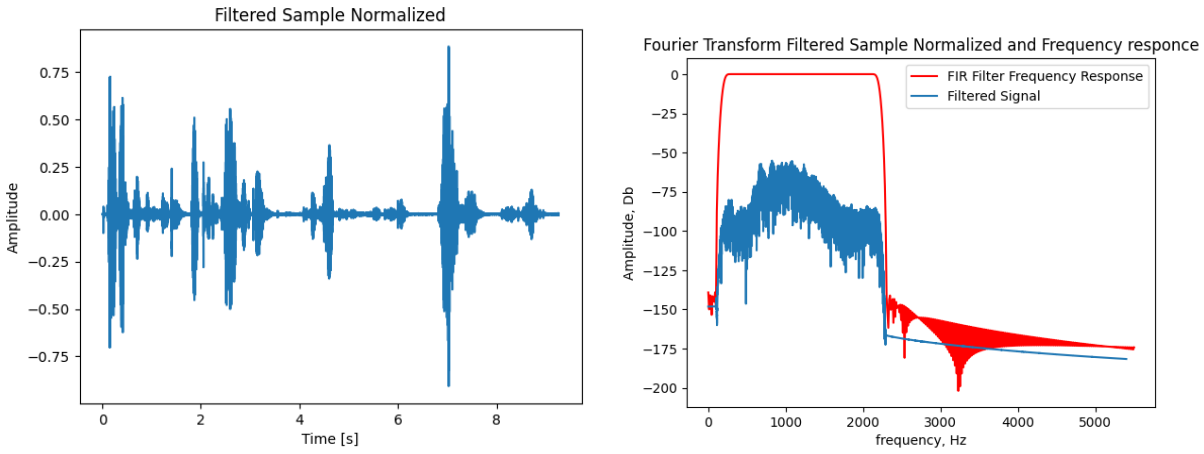
- [**100**] **Hz** : On remarque un pic de fréquence a 100 hz donc notre filtre devra avoir une limite basse supérieur a 100 Hz
- [**200-2200**] **Hz** : Il y a un contenu fréquentiel qui sera sûrement nôtre signal utile. On peux donc limiter la borne supérieur a 2200 Hz
- [**2200-∞**] **Hz** : Ce sont toutes les fréquences qui nous intéresse pas

On va donc dimensionner nôtre filtre passe bande pour grader les valeurs entre 200 et 2200 Hz, Code A.5.

Paramètres fonction firwin() :

1. **Nombre de taps/coefficient** : Pour choisir le nombre de taps in va falloir faire des compromis. En effet, augmenter ceux ci améliore la sélectivité et on a un filtre plus abrupte mais ça augmente aussi le nombre de calcul a réaliser et a garder en mémoire. Donc dans nôtre cas la valeur de 2001 a été le bon compromis temps de calcul vs sélectivité.
2. [**f1, f2**] : Ce sont les deux fréquence limite de notre filtre passe bande.
3. **width** : Ce paramètre va influencer la largeur de la bande de transition entre les bandes de passage et d'arrêt du filtre. Une plus grande largeur de bande de transition peut entraîner une meilleure réponse en fréquence aux dépens de la sélectivité du filtre.

Ensuite, la fonction lfilter est utilisée pour appliquer le filtre FIR conçu au signal original.



(a) Original (b) FFT et réponse en fréquence

FIGURE 3 – Sample Audio Filtrer

On peut remarquer sur l'image 3b qu'on est venu superposer la réponse en fréquence grâce a la fonction freqz du filtre FIR qu'on a créer, Code A.6. Les fréquences hors du filtre sont bien atténuer comme on s'y attend.

3 Conclusion

En conclusion, le filtre FIR est un outil précieux en traitement du signal, offrant une grande adaptabilité aux besoins spécifiques de l'application. Ce rapport a illustré comment concevoir, appliquer et analyser un filtre FIR passe-bande, mettant en lumière son rôle essentiel dans la manipulation des signaux et l'extraction d'informations fréquentielles. L'utilisation judicieuse du filtre FIR peut améliorer de manière significative la qualité et la pertinence de l'analyse de données en temps réel, de la suppression de bruit, de la détection de fréquence et d'autres applications du traitement du signal.

A Python code

A.1 Ajouter librairies

```
import matplotlib.pyplot as plt
import numpy as np
from scipy.io import wavfile
import scipy
from scipy.signal import lfilter, firwin , freqz
```

A.2 Ouvrir fichier audio et extraire infos

```
#Specify the path to your WAV file
file_path = 'speech_three-tones.wav'

#Read the WAV file
samplerate , data = wavfile.read(file_path)
length = data.shape[0] / samplerate

#Print infos of the data
print(f"number of channels = {data.shape[1]}")
print(f"length of sample= {length}s")
print(f"dtypr = {data.dtype}")

#Left Right channel
data_l = data[:, 0]
data_r = data[:, 1]

time = np.linspace(0., length, data.shape[0])
plt.plot(time,data_l/((2**16)/2), label="Left channel")
plt.legend()
plt.xlabel("Time [s]")
plt.ylabel("Amplitude")
plt.show()
```

A.3 Transformation de Fourier sample d'origine

```
# computes the one-dimensional discrete Fourier transform (DFT)
fft_spectrum = np.fft.rfft(data_l)/((2**16)/2)
fft_spectrum_abs = np.abs(fft_spectrum)/data_l.size

freq = np.fft.rfftfreq(data_l.size, d=1./samplerate)

plt.plot(freq[0:50000], 20*np.log10(fft_spectrum_abs[0:50000]))
plt.xlabel("frequency, Hz")
plt.ylabel("Amplitude, Db")
plt.show()")
plt.show()
```

A.4 Filtre FIR

```
#Upper and lower frequency
f1, f2 = 200, 2200

#FIR Window
taps_BP = firwin(2001, [f1, f2],width=200, pass_zero=False,fs=samplerate)

# Use lfilter to filter x with the FIR filter.
filtered_FIR_BP_SP = lfilter(taps_BP, 1.0, data_l)

wavfile.write('filtered_FIR.wav', samplerate, filtered_FIR_BP_SP.astype(np.int16))

plt.title("Filtered Sample Normalized")
plt.plot(time, filtered_FIR_BP_SP/((2**16)/2), label="Left channel")
plt.xlabel("Time [s]")
plt.ylabel("Amplitude")
plt.show()
```

A.5 Transformation de Fourier signal filtré

```
fft_spectrum = np.fft.rfft(filtered_FIR_BP_SP)/((2**16)/2)
freq = np.fft.rfftfreq(filtered_FIR_BP_SP.size, d=1./samplerate)
fft_spectrum_abs = np.abs(fft_spectrum)/filtered_FIR_BP_SP.size

plt.plot(freq[0:50000],20*np.log10(fft_spectrum_abs[0:50000]))
plt.title("Fourier Transform Filtered Sample Normalized")
plt.xlabel("frequency, Hz")
plt.ylabel("Amplitude, Db")
plt.show()
```

A.6 Réponce fréquentiel du filtre

```
from scipy.signal import freqz

# Compute the frequency response of the FIR filter
w, h = freqz(taps_BP, worN=2000)

# Convert angular frequency to Hz
freq_hz = w * samplerate / (2 * np.pi)

# Plot the frequency response
plt.plot(freq_hz[0:500], 20 * np.log10(abs(h))[0:500], 'r', label="FIR Filter Freque
plt.plot(freq[0:50000],20*np.log10(fft_spectrum_abs[0:50000]) ,label="Filtered Signa
plt.title("Fourier Transform Filtered Sample Normalized")
plt.xlabel("frequency, Hz")
plt.ylabel("Amplitude, Db")
plt.legend()
plt.show()
```