1 Rappels

Toutes les questions de ces TDs ne seront pas résolues en séance. Vous êtes invités à essayer de les résoudre (tant que faire se peut ...).

Certains exercices sont issus des cours "discrete-time signal processing" de Oppenheim, Schafer et Buck.

Table de transformées en z

$x[n]$	$X(z)$	Région de convergence (ROC)
$\delta[n]$	1	$z \in \mathbb{C}$
$u[n]$	$\frac{z}{z-1}$	$\lvert z\rvert > 1$
$(-a)^n u(n)$	$\frac{z}{z+a}$	$\lvert z\rvert > a$
$nu[n]$	$\frac{z}{(z-1)^2}$	$\lvert z\rvert > 1$
$n^2 u[n]$	$\frac{z(z+1)}{(z-1)^3}$	$\lvert z\rvert > 1$
$e^{an}u[n]$	$\frac{z}{z-e^a}$	$\lvert z\rvert > \lvert e^a\rvert$
$C_{k-1}^{n-1} e^{a(n-k)}u[n-k]$	$\frac{z}{(z-e^a)^k}$	$\lvert z\rvert > \lvert e^a\rvert$
$\cos(\omega n)u[n]$	$\frac{z(z-\cos(\omega))}{z^2-2z\cos(\omega)+1}$	$\lvert z\rvert > 1$
$\sin(\omega n)u[n]$	$\frac{z\sin(\omega)}{z^2-2z\cos(\omega)+1}$	$\lvert z\rvert > 1$
$\frac{1}{n}u[n-1]$	$\ln\left(\frac{z}{z-1}\right)$	$\lvert z\rvert > 1$
$\sin(\omega n+\theta)u[n]$	$\frac{z^2\sin(\theta)+z\sin(\omega-\theta)}{z^2-2z\cos(\omega)+1}$	$\lvert z\rvert > 1$
$e^{an}\cos(\omega n)u[n]$	$\frac{z(z-e^a\cos(\omega))}{z^2-2ze^a\cos(\omega)+e^{2a}}$	$\lvert z\rvert > \lvert e^a\rvert$ height

1.1 Réponses fréquentielles (en amplitude et en phase)

Calculez et tracez les réponses fréquentielles des systèmes décrits par les équations aux différences suivantes :

- $y[n] = x[n] + 2x[n-1] + 3x[n-2] + 2x[n-3] + x[n-4]$
- $y[n] = y[n-1] + x[n]$
- $y[n] = x[n] + 3x[n-1] + 2x[n-2]$

1.2 Système LTI

Soit le filtre numérique exprimé par

$$H(z) = \frac{1 + 2.05.z^{-1} - 2.85.z^{-2}}{(1 + 0.9z^{-2})(1 + 0.95z^{-1})}$$

Quel est le type de ce filtre (IIR/FIR). Donnez son équation de différences.

Trouvez les pôles et les zéros et représentez-les sur le plan complexe.

Donnez l'expression de $H(f)$.

Tracez le gain du filtre $|H(f)|$ entre $f = -/2$ et $f = 1/2$, pour 5 valeurs positives de f.

Justifiez ce tracé en utilisant le diagramme des pôles et zéros.

1.3 Conception d'un filtre simple

Concevez un système LTI causal (à temps discret) avec les propriétés suivantes :

- Le système doit préserver exactement le signal $\cos(0.5\pi n)$.
- Le système doit annihiler les signaux constants (réponse fréquentielle nulle à la fréquence 0).

Pour ce système, donnez l'équation aux différences (il est possible d'avoir une réponse impulsionnelle de longueur 3), tracez la réponse fréquentielle, le diagramme des pôles et zéros.

1.4 Paramètres d'un filtre

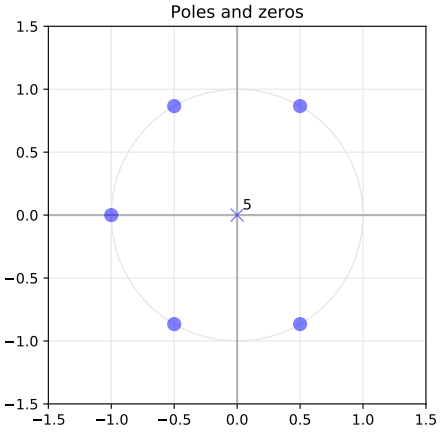
Soit un filtre donné par la fonction de transfert suivante :

$$H(z) = \frac{\delta_o + \delta_1 z^{-1} - \delta_2 z^{-1} f(z)}{1 - (1 + m_1)z^{-1} - m_2 z^{-1} f(z)}$$

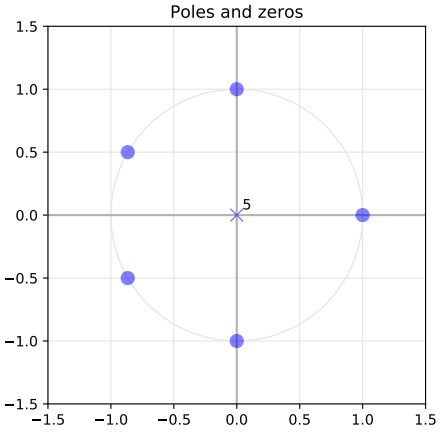
où $f(z) = \frac{1}{1-z^{-1}}$. Faites la conception du filtre de telle sorte qu'il ait un gain unitaire en courant continu et un zéro double (deux zéros) à $\omega = \pi$.

1.5

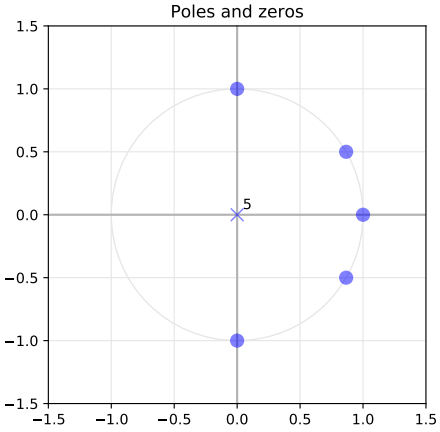
Soient les 6 filtres (FIR) suivants, donnez les correspondances entre les figures (pole/zéro - réponse impulsionnelle - réponse fréquentielle) ,


Poles and zeros
5

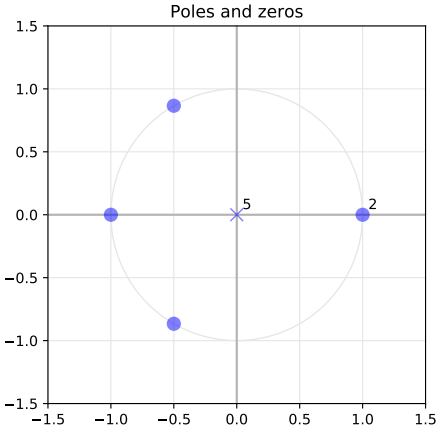
(1)


Poles and zeros
5

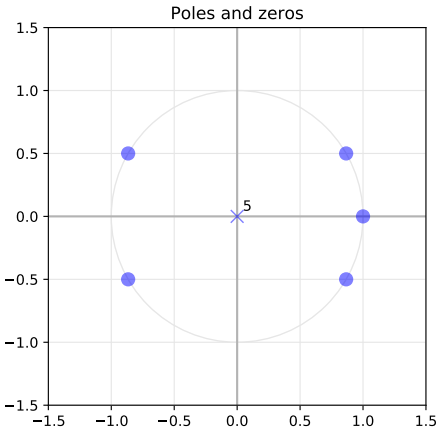
(2)


Poles and zeros
5

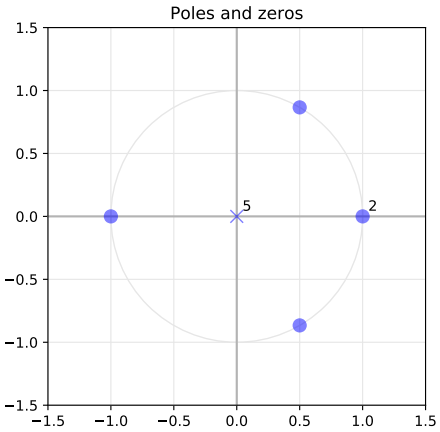
(3)


Poles and zeros
5
2

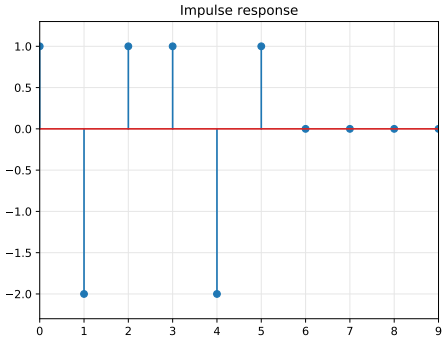
(4)


Poles and zeros
5

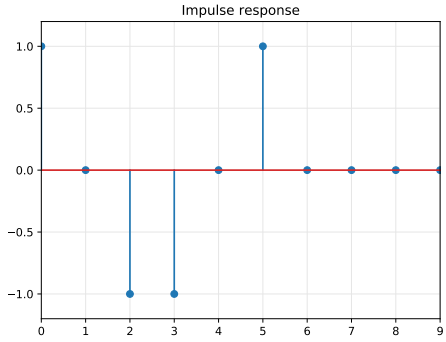
(5)


Poles and zeros
5
2

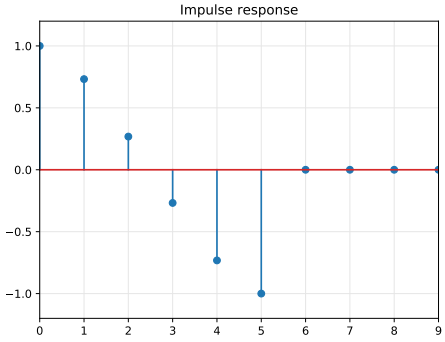
(6)


Impulse response

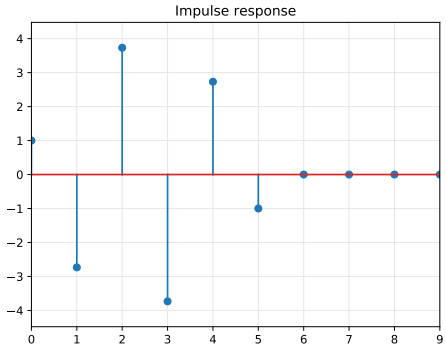
(1)


Impulse response

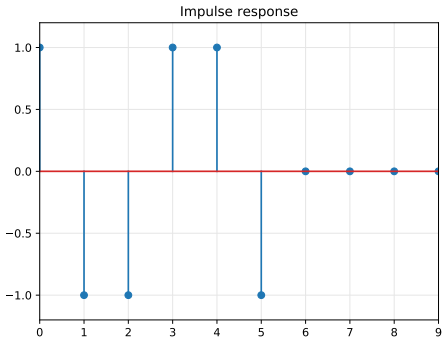
(2)


Impulse response

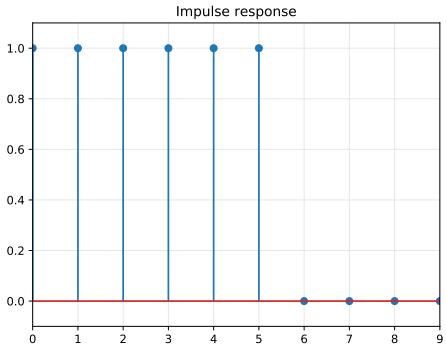
(3)


Impulse response

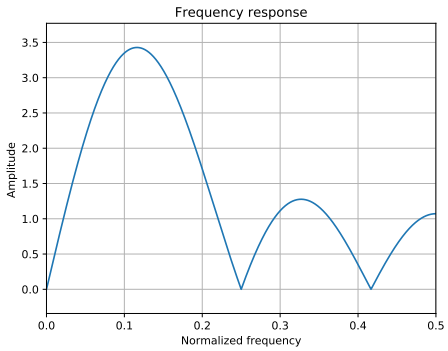
(4)


Impulse response

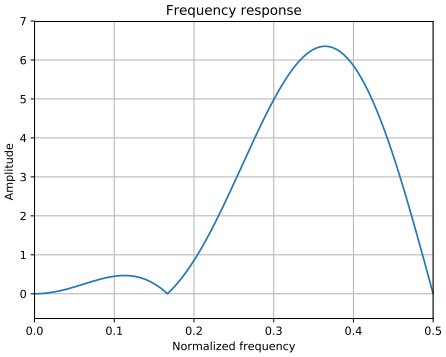
(5)


Impulse response

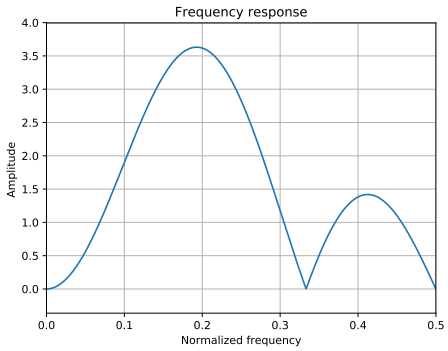
(6)


Frequency response
Amplitude
Normalized frequency

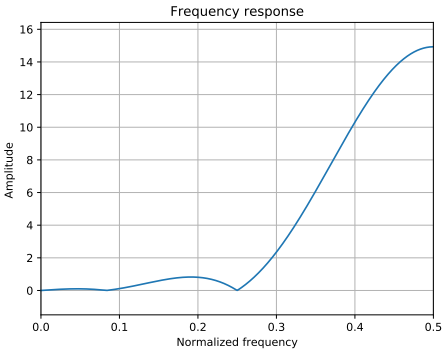
(1)


Frequency response
Amplitude
Normalized frequency

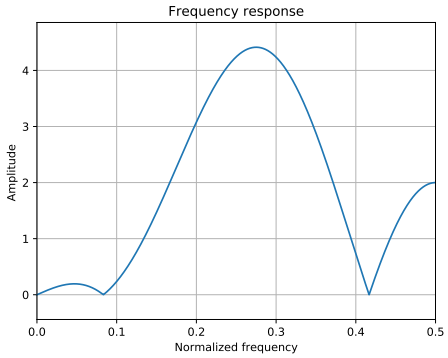
(2)


Frequency response
Amplitude
Normalized frequency

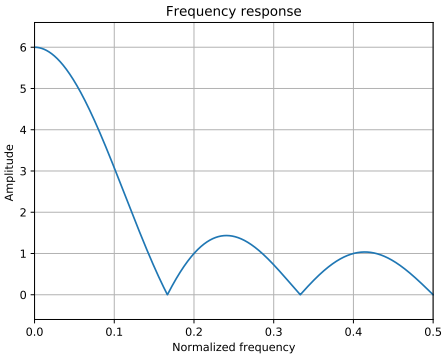
(3)


Frequency response
Amplitude
Normalized frequency

(4)


Frequency response
Amplitude
Normalized frequency

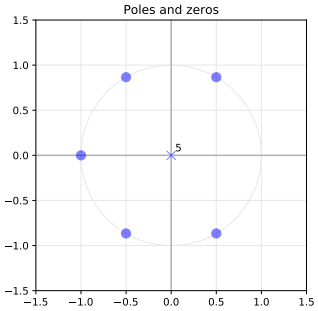
(5)


Frequency response
Amplitude
Normalized frequency

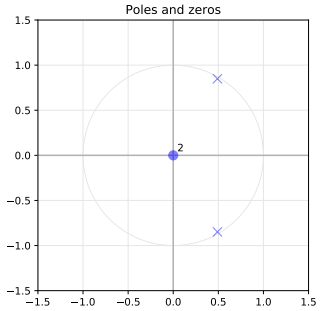
(6)

1.6

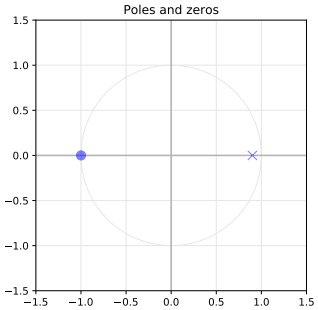
Soient les 6 filtres suivants, donnez les correspondances entre les figures (pole/zéro - réponse impulsionnelle - réponse fréquentielle) ,



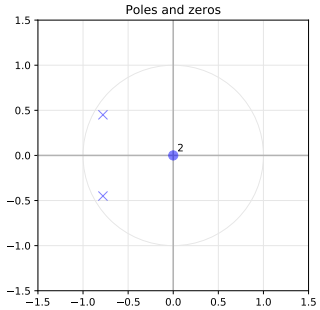
(1)



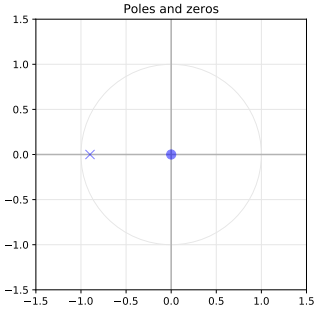
(2)



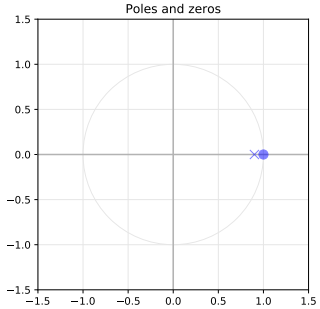
(3)



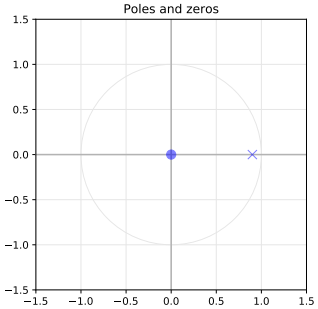
(4)



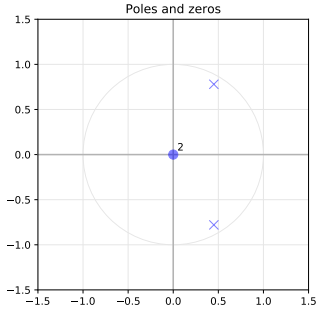
(5)



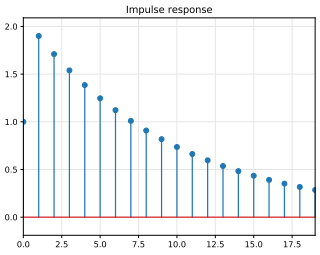
(6)



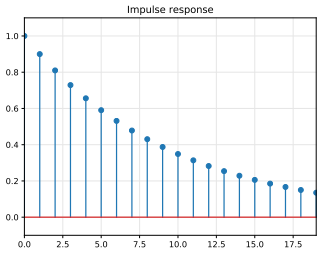
(7)



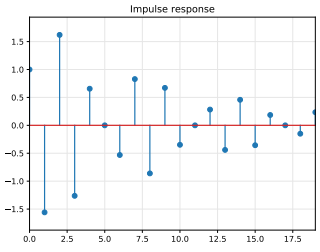
(8)


Impulse response

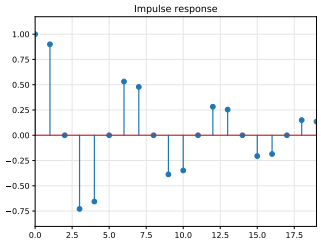
(1)


Impulse response

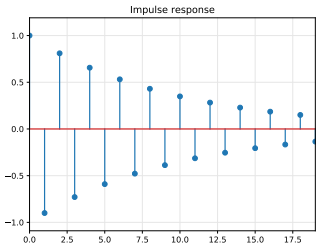
(2)


Impulse response

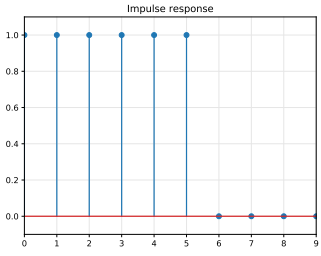
(3)


Impulse response

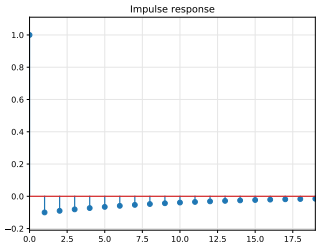
(4)


Impulse response

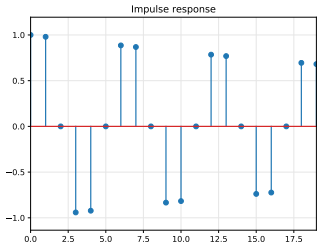
(5)


Impulse response

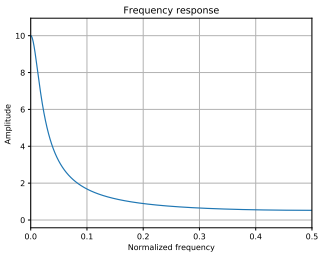
(6)


Impulse response

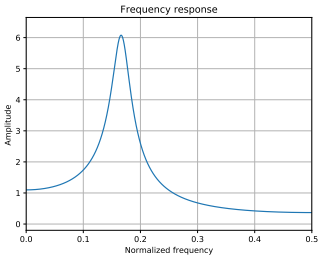
(7)


Impulse response

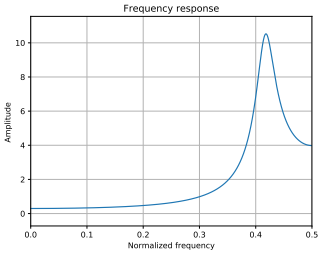
(8)


Frequency response
Amplitude
Normalized frequency

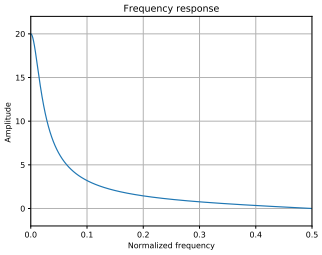
(1)


Frequency response
Amplitude
Normalized frequency

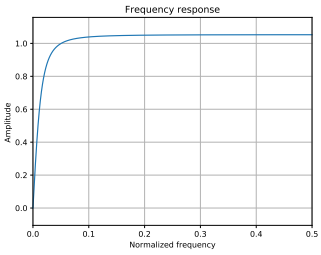
(2)


Frequency response
Amplitude
Normalized frequency

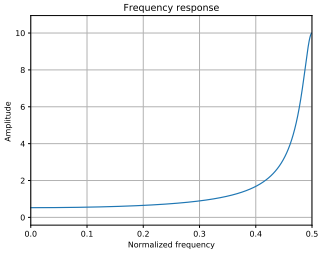
(3)


Frequency response
Amplitude
Normalized frequency

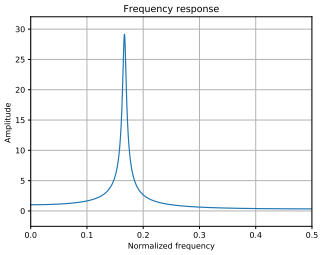
(4)


Frequency response
Amplitude
Normalized frequency

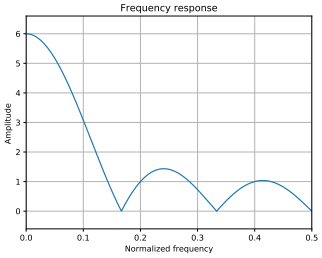
(5)


Frequency response
Amplitude
Normalized frequency

(6)


Frequency response
Amplitude
Normalized frequency

(7)


Frequency response
Amplitude
Normalized frequency

(8)

2 Conception de filtres FIR

Les exercices suivant sont à faire avec Python (sur Spyder ou Jupyter).

2.1 Comparaison des fenêtres

Concevez un ensemble de filtres passe-bas de fréquence de coupure $\omega_c = \pi/4$, d'ordre 11. Vous utiliserez les fenêtres rectangulaire, de Hamming, de Hanning, de Blackmann et de Blackmann Harris. Comparez les différentes caractéristiques de ces filtres.

Modifiez l'ordre du filtre, quel en est l'effet ?

Exprimez les pentes des zones de transition en dB/octave ou dB/décade.

2.2 Elimination du bourdonnement à 100 Hz

Un problème classique dans les enregistrements audio de mauvaise qualité, où dans les transmissions sur supports analogiques, est la présence d'un bourdonnement à 50 Hz. Pour l'exercice nous utiliserons un bourdonnement à 100 Hz.

Dans un premier temps, vous allez charger un fichier musical de votre choix, le décimer (passer de 44 kéch/sec à 5.5 kéchs/sec), additionner un signal sinusoïdal à 100 Hz et ... écouter le résultat.

Dans un deuxième temps, vous allez concevoir un filtre permettant d'éliminer ce bruit, en essayant de détériorer le signal original le moins possible. Comparez les deux résultats.

2.3 Elimination du battement

Quand on additionne deux signaux sinusoïdaux de fréquence proche, apparaît un battement. En effet :

$s(t) = \sin(\omega_1 t) + \sin(\omega_2 t) = 2\sin(\frac{\omega_1 + \omega 2}{2}t)\cos(\frac{\omega_1 - \omega 2}{2}t)$

et donc $s(t)$ est un signal à la moyenne des fréquences, modulé par un signal à la (moitié de la) différence des fréquences, appelé battement.

Dans cet exercice, on travaillera dans un premier temps avec une $f_1 = 880Hz, f_2 = 884Hz$. On déterminera un ordre de filtre FIR par "kaiserord", et on fera le design du filtre avec l'algorithme de Remez.

Commentez la longueur du filtre, et donnez sa complexité (en nombre de multiplications par seconde nécessaires).

Exemple de code pour générer un battement :

```
%matplotlib inline
import numpy as np
import scipy.signal as signal
import matplotlib.pyplot as plt
import numpy as np
import IPython



Fs=8000

f1=880
```

```
f2=884
om1=2*np.pi*f1
om2=2*np.pi*f2
t=np.arange(0,5,1/Fs)
y=np.sin(om1*t)+np.sin(om2*t)


IPython.display.Audio(y, rate=Fs)
```

3 IIR filter design

3.1 IIR vs FIR Exercice

This computer exercice is devoted to the comparison between IIR and FIR filters. In particular, you will explore the different type of IIR filters (Bessel, Elliptic, Butterworth, Chebyshev I and II), and draw a table with the advantages of each of these filters.

You will also compare the computation complexity (expressed in multiplications per sample, or multiplications per seconds) between IIR and FIR filters.

To develop your comparisons, you will design a collection of Low Pass filters (conclusions would be similar for High Pass, Bandpass and Bandstop filters).

The parameters will be :

> Sampling Frequency : Fs=1000HzFs=1000Hz First cutoff frequency Fc1=100HzFc1=100Hz Three different second cutoff frequency Fc2=110,125,200HzFc2=110,125,200Hz Three different In-band Ripple : 0.01, 0.1, 1 dB Two different Out-of Band attenuations : 30 and 80 dB

For these cases, you will design a FIR filter (estimate the order with kaiserord), and then design the different IIR filters. From these results, you will develop your comparisons and write a short tutorial whose title will be "In which case to use specific IIR filters".

To help you, you will find hereunder a function to plot filter templates (the function is documented), and an example of an FIR vs IIR filter.

3.2 Eliminate Beat Frequency

In the previous computer exercice, you were asked to eliminate beat frequency occuring when you have two close frequencies (here f_1=440Hz and f_2=442Hz.).

Based on the conclusions drawn when answering the previous question, select an IIR filter and design a filter eliminating the 442 Hz frequency. What is the ratio between the computational complexity of the FIR filter and the IIR filter ?

A Notebook with a bunch of code is provided on Moodle.

4 Homework 1

L'objectif de ce devoir est de comparer les filtres FIR et IIR dans le cas de débruitage de signaux.

4.1 Note sur les valeurs de "Ripple" en design FIR et IIR

Le ripple (ondulation), pour les filtres IIR, est spécifié de telle sorte que le gain maximum vaut 1 (0 dB), et donc, dans la bande passante, le gain est compris entre 0 dB et -δ_{IIR} dB ($10^{-\delta_{\text{IIR}}/20}$). Dans le cas des filtres FIR (algorithme de Remez), il est spécifié entre $1 + \delta_{\text{FIR}}$ et $1 - \delta_{\text{FIR}}$ (en naturel). Il faut que vous en teniez compte pour avoir les mêmes gains moyens dans la bande passante, et donc multiplier (par exemple le IIR) par un facteur qui dépend de ces valeurs de ripple.

4.2 Débruitage

L'objectif des filtres sera de débruiter un signal donné, dans ce cas-ci un signal audio, en utilisant une fréquence d'échantillonnage de 44100 Hz.

Dans un premier temps, vous fabriquerez trois types de bruit : un bruit basse fréquences, un bruit passe-bande et un bruit haute fréquence. Vous pouvez choisir les bandes de fréquences qui vous paraissent raisonnables (la bande que j'ai prise dans l'exemple ci-dessous est ... très étroite). Pour la génération de ce bruit, inspirez vous du code ci-dessous.

Ensuite, vous prendrez un signal audio existant, ou en fabriquerez un vous même (par exemple un chirp, qui balaie toutes les fréquences de 0 à 22050 Hz), et l'affecterez du bruit généré précédemment.

Ensuite, vous concevrez les filtres permettant de débruiter le signal. Vous utiliserez les filtres FIR de Remez et de Kaiser, ainsi que les filtres IIR elliptiques, de Chebyshev I et II et de Butterworth, et les appliquerez au signal bruité.

Pour chacun de ces filtres vous fournirez :

- Les spécifications de vos filtres (bandes passantes, ripple, atténuations), les gabarits correspondants et les caractéristiques du filtre (ordre, type de filtre, amplitude de la réponse fréquentielle en dB, linéarité de la phase, le diagramme zéros-pôles pour les IIR).
- Le nombre de multiplications nécessaires par seconde (tenir compte de la symétrie des FIRs).
- discutez de la performance des filtres en étudiant le SNR après débruitage (et éventuellement sur base de votre écoute des signaux).

En fonctions de ces résultats, donnez des recommandations pour le débruitage de signaux audio.

4.3 Exemple de calcul de SNR après débruitage

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
```

```
Created on Tue Nov 27 23:40:50 2018

@author: ld
"""


import scipy.io.wavfile
import sounddevice as sd
import numpy as np
import scipy.signal as signal
import matplotlib.pyplot as plt


fs,y=scipy.io.wavfile.read('piano.wav')
y=y.astype(float)/np.max(y)    #cast y to float and normalize between -1 and 1
y=y/2                          # normalize between -0.5 and 0.5, such that nois
                               #  does not cause noisy signal to exceed 1.0
y=y[0:fs]                      # only take one second

Sy=np.linalg.norm(y)**2        # This is the power (energy too) of y

                               # generate white noise
noise=np.random.normal(0,1,np.size(y))
                               # filter it between 220 and 250 Hz
firnoise=signal.remez(5501,np.array([0,200,220,250,270,fs/2])/
    ... fs,np.array([0.0,1.0,0.0]),type='bandpass')
                               # plot the response of the filter
w, Hk = signal.freqz(firnoise,1)
plt.plot(w/(2*np.pi)*fs, 20*np.log10(np.abs(Hk)))
plt.show()

                               # actually filter the noise
filtered_noise=2*np.convolve(firnoise,noise)
                  # add the noise to the signal (take group delay into account
ynoisy=y+filtered_noise[2750:y.size+2750:]

                            # design (crudely) the denoising filter
firdenoise=signal.remez(5501,np.array([0,200,210,260,270,fs/2])/
    ... fs,np.array([1.0,0.0,1.0]),type='bandpass')

ydenoise=np.convolve(firdenoise,ynoisy)
ydenoise=ydenoise[2750:y.size+2750:]  # take the group delay into account

# check the sound
sd.play(ydenoise,fs)

# draw the different power spectra
f,Pyy_den=signal.periodogram(y,fs)
plt.semilogy(f, Pyy_den)
plt.ylim([1e-9, 1])
```

```
plt.xlabel('frequency [Hz]')
plt.ylabel('PSD [V**2/Hz]')
plt.title('power spectrum of the input')
plt.show()

f,Pyy_den=signal.periodogram(ynoisy,fs)
plt.semilogy(f, Pyy_den)
plt.ylim([1e-9, 1])
plt.xlabel('frequency [Hz]')
plt.ylabel('PSD [V**2/Hz]')
plt.title('power spectrum of the noisy signal')

plt.show()
f,Pyy_den=signal.periodogram(ydenoise,fs)
plt.semilogy(f, Pyy_den)
plt.ylim([1e-9, 1])
plt.xlabel('frequency [Hz]')
plt.ylabel('PSD [V**2/Hz]')
plt.title('power spectrum of the denoised signal')

plt.show()

print('power of y = ',20*np.log10(np.linalg.norm(y)), 'dB')
print('SNR before denoising = ',20*np.log10(np.linalg.norm(y)/
    ... np.linalg.norm(y-ynoisy)),' dB')
print('SNR after denoising = ',20*np.log10(np.linalg.norm(y)/
    ... np.linalg.norm(y-ydenoise)),' dB')
```