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) ,

$$h[n] = \delta(n) + 2\delta(n-1) + 3\delta(n-2) + 2\delta(n-3) + \delta(n-4)$$

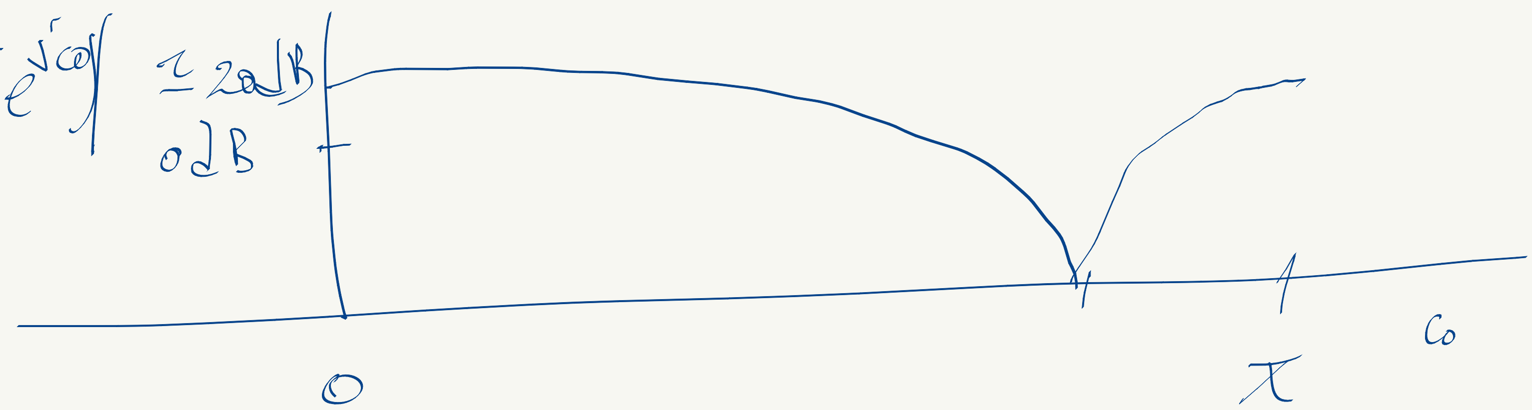
$$H(e^{j\omega}) = e^{-2j\omega}\left[e^{2j\omega} + 2e^{j\omega} + 3 + 2e^{-j\omega} + e^{-2j\omega}\right]$$

$$= e^{-2j\omega}\left[3 + 4\cos\omega + 2\cos 2\omega\right]$$

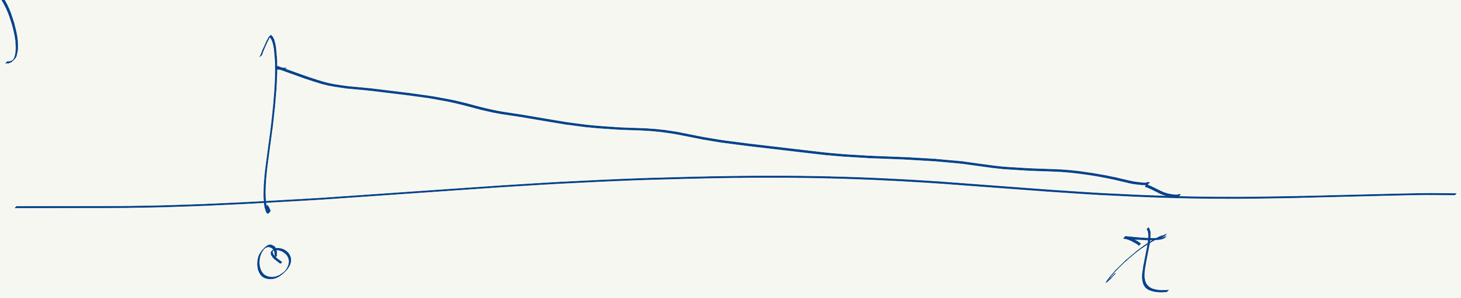
$$|H(e^{j\omega})| = |3 + 4\cos\omega + 2\cos 2\omega|$$

$$\angle H(e^{j\omega}) = -2\omega$$

$20\log_{10}|H(e^{j\omega})|$ $\approx 20\,dB$



$\angle H(e^{j\omega})$



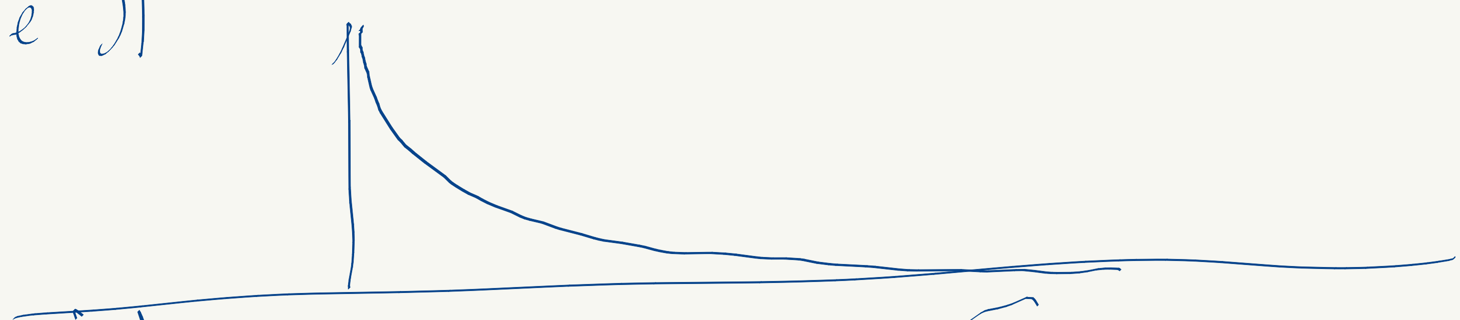
$$Y(z) = z^{-1}Y(z) + X(z)$$

$$\Rightarrow H(z) = \frac{Y(z)}{X(z)} = \frac{1}{1-z^{-1}} = z^{1/2}\,\frac{1}{z^{1/2}-z^{-1/2}}$$

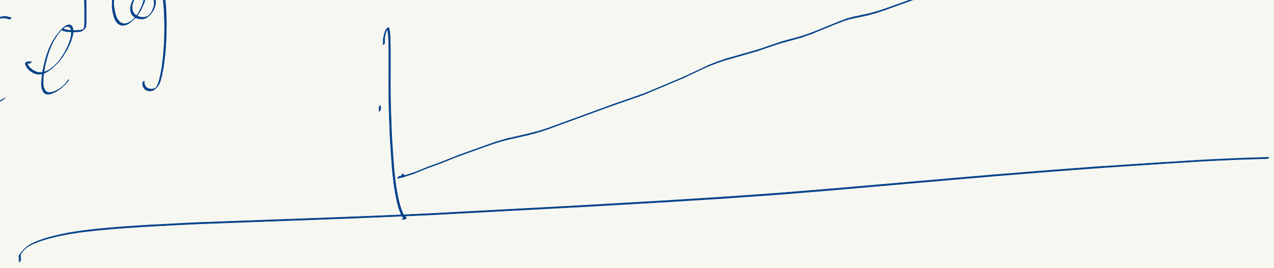
$$|H(e^{j\omega})| = \frac{1}{\sqrt{(1-\cos\omega)^2+\sin^2\omega}} = \frac{1}{\sqrt{2-2\cos\omega}}$$

$$\angle H(e^{j\omega}) = -\operatorname{atan}\frac{\sin\omega}{1-\cos\omega} = \omega/2 - \pi/2$$

$20\log_{10}|H(e^{j\omega})|$



$\angle H(e^{j\omega})$



$$h[n] = \delta[n] + 3\delta[n-1] + \delta[n-2]\cdot 2$$

$$H(e^{j\omega}) = 1 + 3e^{-j\omega} + 2e^{-j2\omega}$$

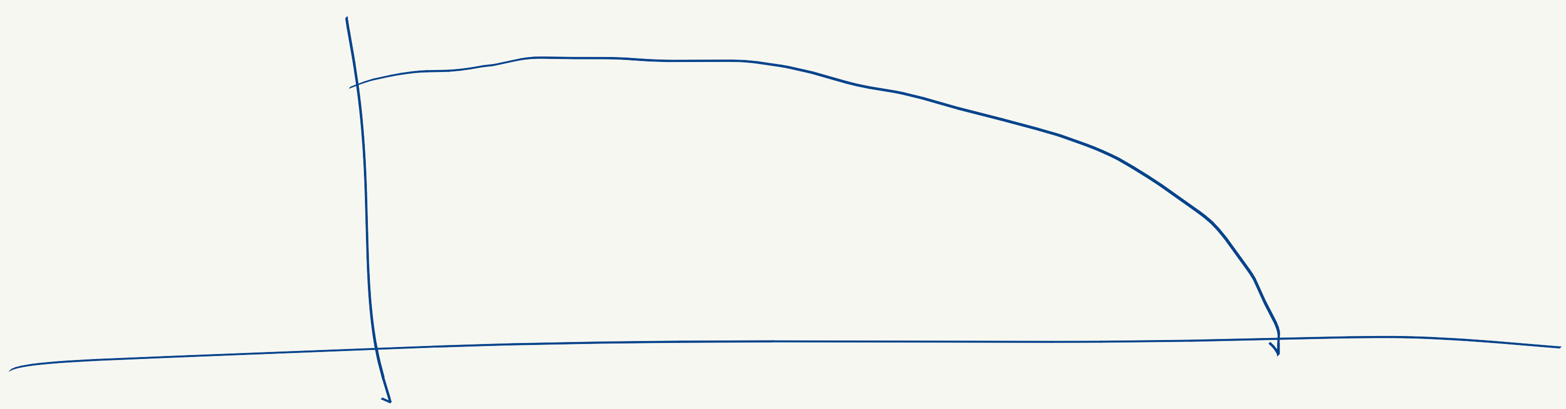
$$= 2\cos\frac{\omega}{2}\left[e^{-j\frac{\omega}{2}} + 2e^{-j\frac{3\omega}{2}}\right]$$

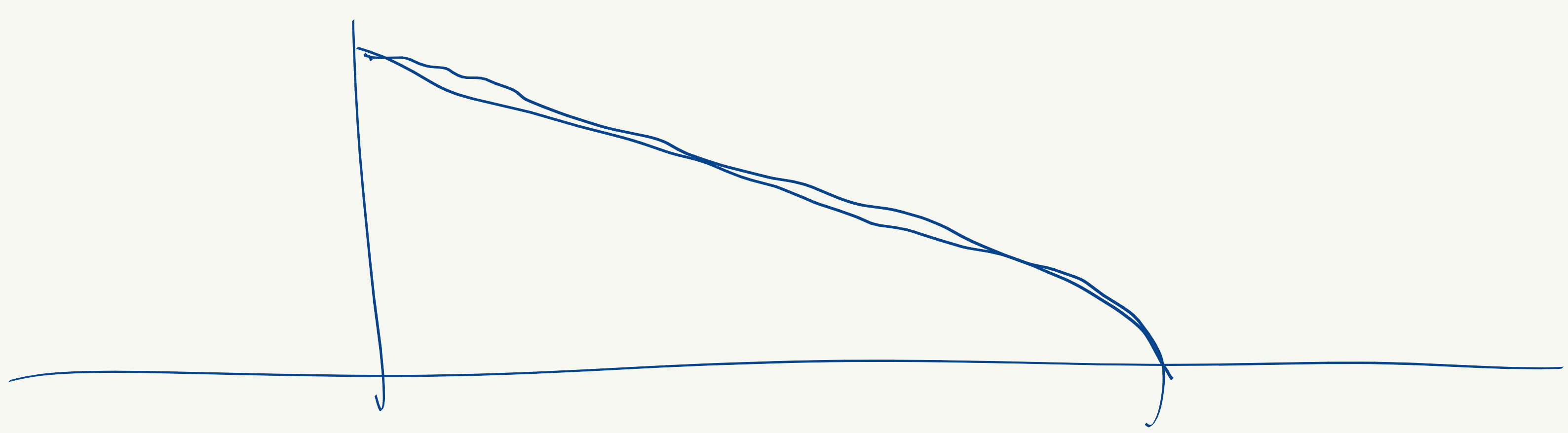
$$= 2e^{-j\frac{\omega}{2}}\cos\frac{\omega}{2}\left[1 + 2e^{-j\omega}\right]$$

$$|H(e^{j\omega})| = 2\left|\cos\frac{\omega}{2}\right|\sqrt{(1+2\cos\omega)^2 + 4\sin^2\omega}$$

$$= 2\left|\cos\frac{\omega}{2}\right|\sqrt{5 + 4\cos\omega}$$

$$\angle H(e^{j\omega}) = -\frac{\omega}{2} + \arctan\frac{2\sin\omega}{1+2\cos\omega}$$





1.4 $$H(z) = \frac{\delta_0(1-z^{-1}) + \delta_1(1-z^{-1})z^{-1} - \delta_2 z^{-1}}{1-z^{-1} - (1+m_1)z^{-1}(1-z^{-1}) - m_2 z^{-1}}$$

$$= \frac{\delta_0 - z^{-1}(\delta_0 - \delta_1 + \delta_2) - \delta_1 z^{-2}}{1 - z^{-1}(2+m_1+m_2) + (1+m_1) z^{-2}}$$

(1) DC : $z = 1$

$$H(z=1) = \frac{\delta_0 - (\delta_0 - \delta_1 + \delta_2) - \delta_1}{1 - (2+m_1+m_2) + (1+m_1)} = \frac{-\delta_2}{-m_2}$$

$$\Rightarrow \delta_2 = +m_2$$

(2) $\omega = \pi \rightarrow z = -1$ → il faut un zéro double au numérateur

→ vérifier que ce n'est pas un zéro du dénominateur !

$\Rightarrow N(z)$ de la forme

$$N(z) = \delta_0(z^{-1}+1)^2$$

$$N(z) = \delta_0 (z^{-1}+1)^2 = \delta_0 + 2\delta_0 z^{-1} + \delta_0 z^{-2}$$

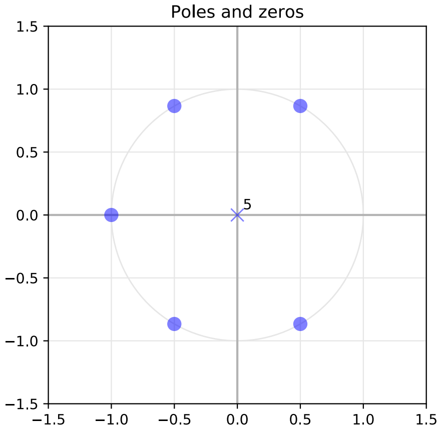
$$\Rightarrow \delta_0 - \delta_1 + \delta_2 = -2\delta_0$$

$$\delta_1 = -\delta_0$$

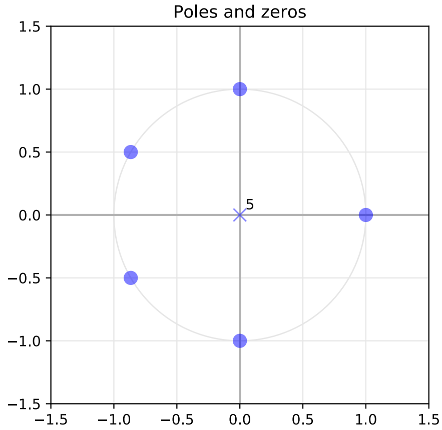
$$\delta_0 + \delta_0 + \delta_2 = -2\delta_0$$

$$\Rightarrow \delta_2 = -4\delta_0$$

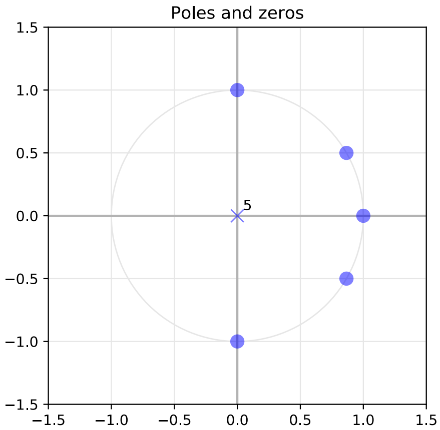
$$\Rightarrow \delta_0 = \frac{m_2}{4} \qquad \delta_1 = \frac{m_2}{4}$$


Poles and zeros
5

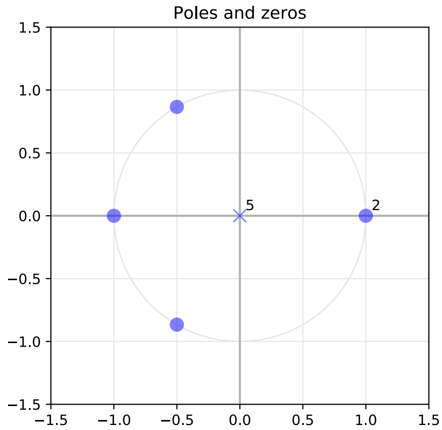
(1)


Poles and zeros
5

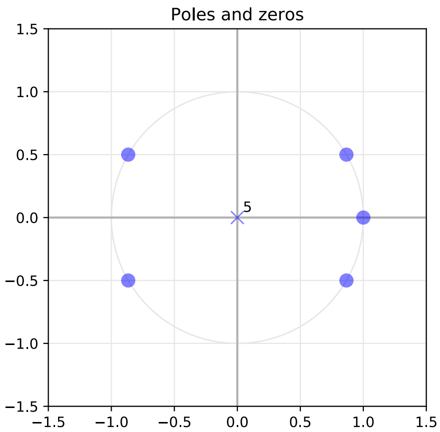
(2)


Poles and zeros
5

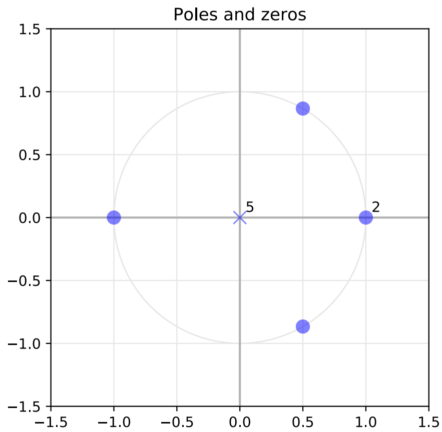
(3)


Poles and zeros
5
2

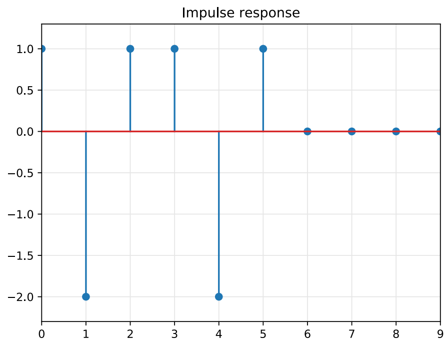
(4)


Poles and zeros
5

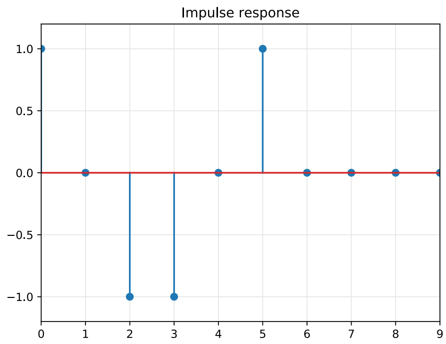
(5)


Poles and zeros
5
2

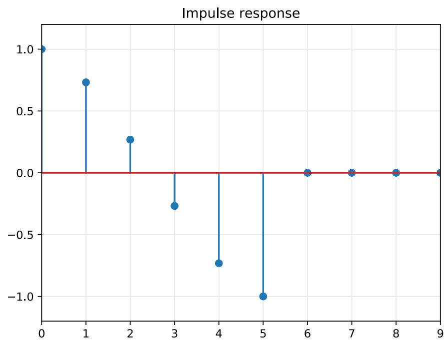
(6)


Impulse response

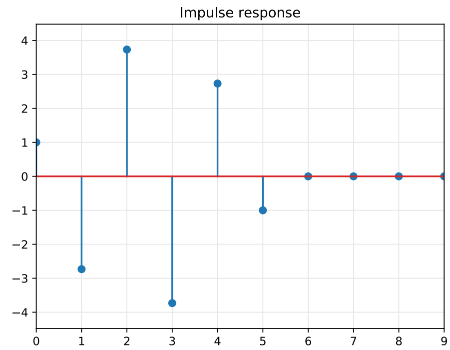
(1)


Impulse response

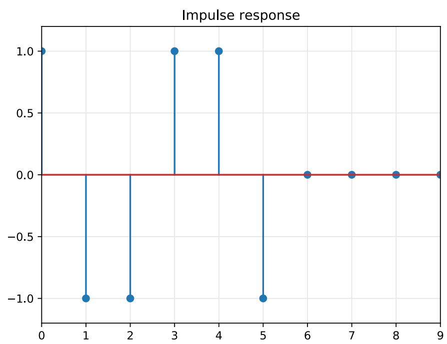
(2)


Impulse response

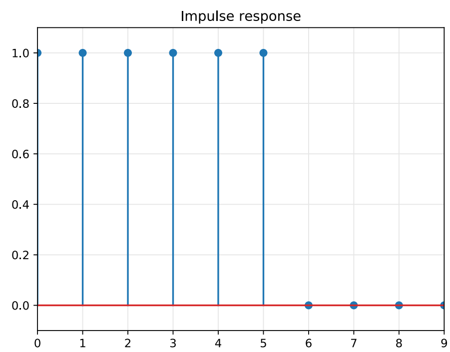
(3)


Impulse response

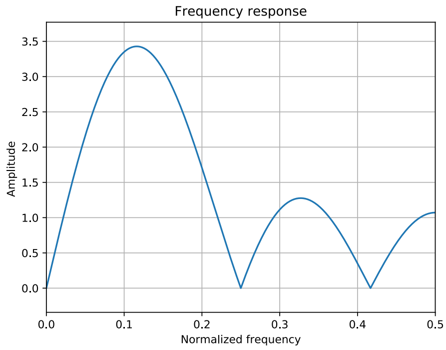
(4)


Impulse response

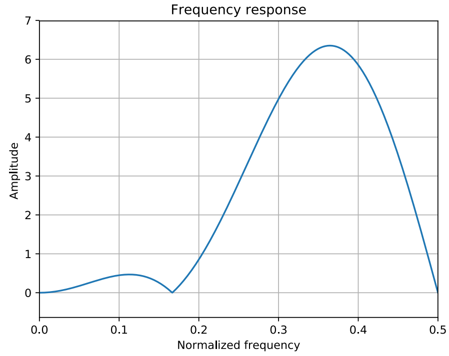
(5)


Impulse response

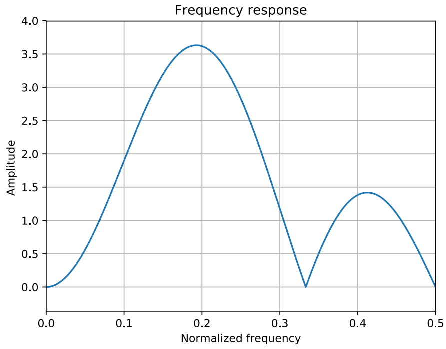
(6)


Frequency response
Amplitude
Normalized frequency
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
0.0
0.1
0.2
0.3
0.4
0.5

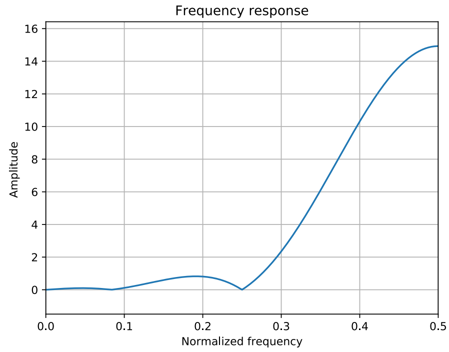
(1)


Frequency response
Amplitude
Normalized frequency
0
1
2
3
4
5
6
7
0.0
0.1
0.2
0.3
0.4
0.5

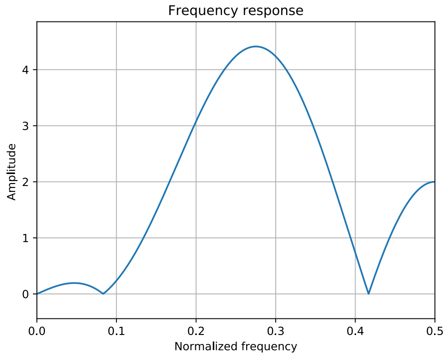
(2)


Frequency response
Amplitude
Normalized frequency
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
0.0
0.1
0.2
0.3
0.4
0.5

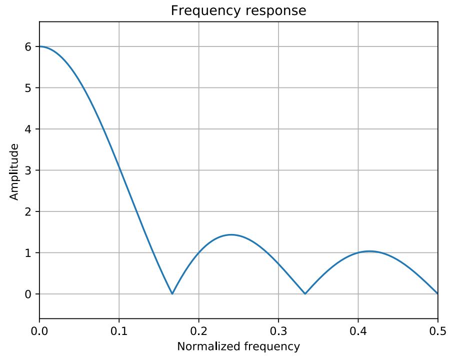
(3)


Frequency response
Amplitude
Normalized frequency
0
2
4
6
8
10
12
14
16
0.0
0.1
0.2
0.3
0.4
0.5

(4)


Frequency response
Amplitude
Normalized frequency
0
1
2
3
4
0.0
0.1
0.2
0.3
0.4
0.5

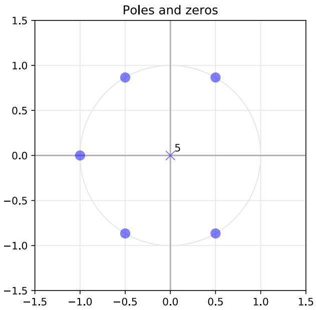
(5)


Frequency response
Amplitude
Normalized frequency
0
1
2
3
4
5
6
0.0
0.1
0.2
0.3
0.4
0.5

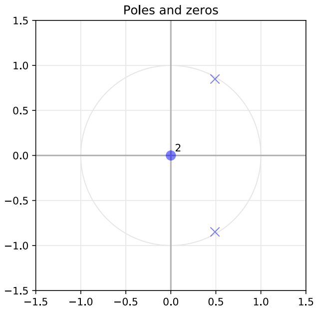
(6)

1.6

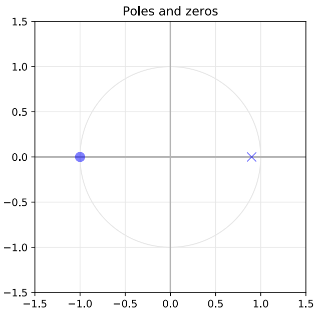
Soient les 6 filtres suivants, donnez les correspondances entre les figures (pole/zéro - réponse impulsionnelle - réponse fréquentielle) ,



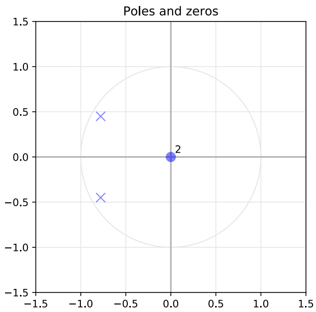
(1)



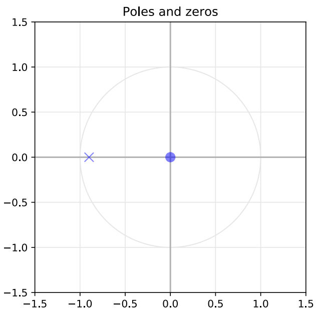
(2)



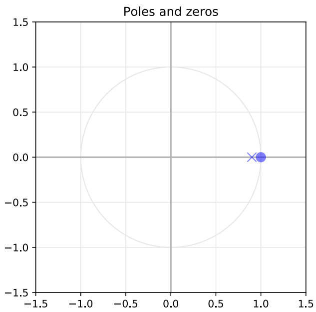
(3)



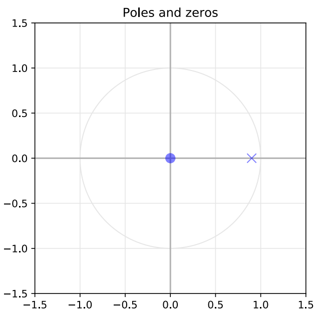
(4)



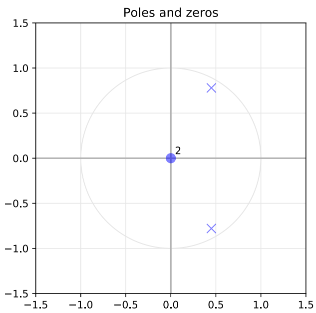
(5)



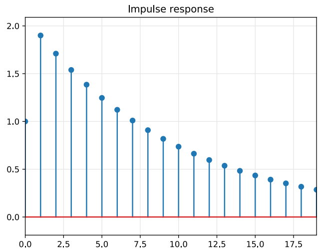
(6)



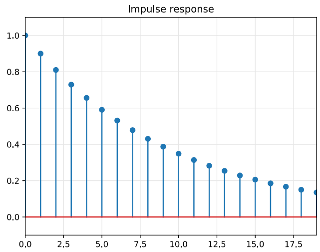
(7)



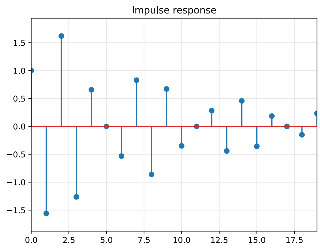
(8)


Impulse response

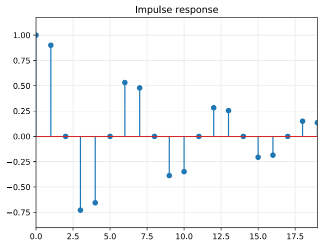
(1)


Impulse response

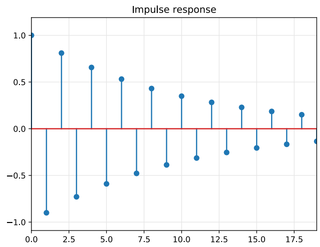
(2)


Impulse response

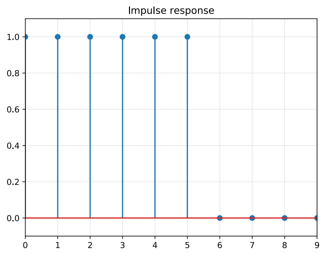
(3)


Impulse response

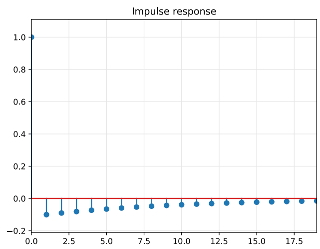
(4)


Impulse response

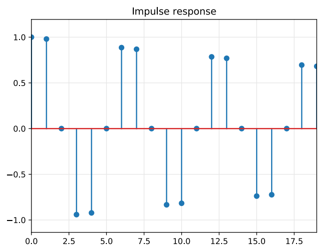
(5)


Impulse response

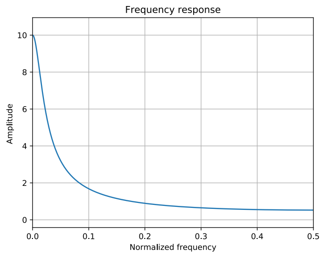
(6)


Impulse response

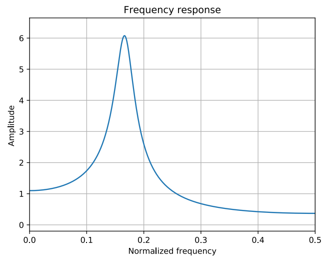
(7)


Impulse response

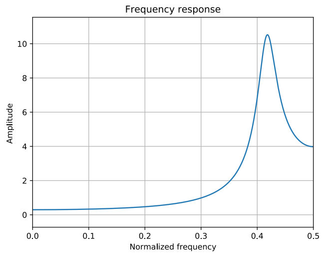
(8)


Frequency response
Amplitude
Normalized frequency
0 2 4 6 8 10
0.0 0.1 0.2 0.3 0.4 0.5

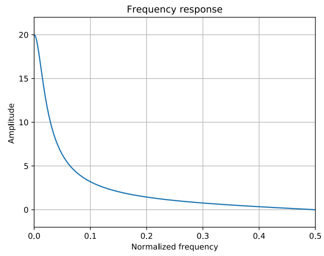
(1)


Frequency response
Amplitude
Normalized frequency
0 1 2 3 4 5 6
0.0 0.1 0.2 0.3 0.4 0.5

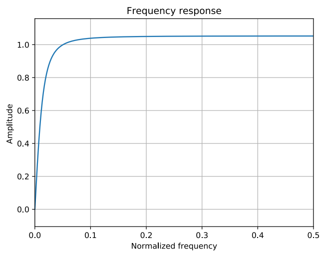
(2)


Frequency response
Amplitude
Normalized frequency
0 2 4 6 8 10
0.0 0.1 0.2 0.3 0.4 0.5

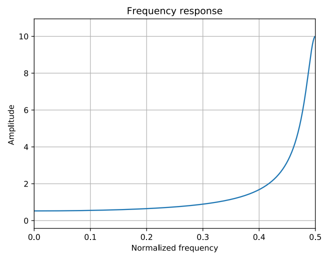
(3)


Frequency response
Amplitude
Normalized frequency
0 5 10 15 20
0.0 0.1 0.2 0.3 0.4 0.5

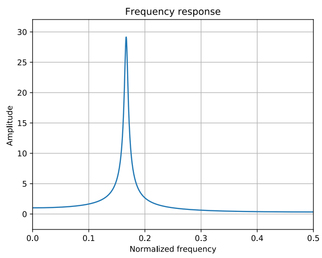
(4)


Frequency response
Amplitude
Normalized frequency
0.0 0.2 0.4 0.6 0.8 1.0
0.0 0.1 0.2 0.3 0.4 0.5

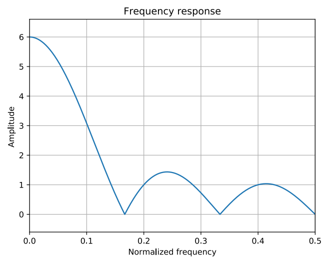
(5)


Frequency response
Amplitude
Normalized frequency
0 2 4 6 8 10
0.0 0.1 0.2 0.3 0.4 0.5

(6)


Frequency response
Amplitude
Normalized frequency
0 5 10 15 20 25 30
0.0 0.1 0.2 0.3 0.4 0.5

(7)


Frequency response
Amplitude
Normalized frequency
0 1 2 3 4 5 6
0.0 0.1 0.2 0.3 0.4 0.5

(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')
```

5 Changement de fréquence d'échantillonnage

5.1 Echantillonnage et filtrage

Soit un signal réel $x_c(t)$ à temps continu, avec le support spectral $\Omega = [-2\pi 5.10^3, 2\pi 5.10^3]$. Ce signal est échantillonné à une fréquence d'échantillonnage $\dfrac{1}{T_1}$. Ensuite, il passe dans un filtre de réponse fréquentielle :

$$H(e^{j\omega}) = \begin{cases} 1 & |\omega| \leq \frac{\pi}{2} \\ 0 & \frac{\pi}{2} < |\omega| < \pi \end{cases}$$

Ensuite, le signal passe dans un convertisseur temps discret/temps continu idéal, en supposant que l'intervalle entre deux échantillons vaut T_2 pour donner le signal $y_c(t)$.

Tracez les graphes de $Y_C(j\Omega)$ pour :

- $\frac{1}{T_1} = \frac{1}{T_2} = 10^4$
- $\frac{1}{T_1} = \frac{1}{T_2} = 2.10^4$
- $\frac{1}{T_1} = 2.10^4$; $\frac{1}{T_2} = 10^4$
- $\frac{1}{T_1} = 10^4$; $\frac{1}{T_2} = 2.10^4$

5.2 Changement de fréquence d'échantillonnage

Soit un signal de parole $x_c(t)$ échantillonné à 10 kéchs/seconde, l'échantillonneur étant précédé d'un filtre anti-repli idéal. On notera le signal échantillonné $x_1[n]$. Ce même signal de parole $x_c(t)$ est échantillonné à 6 kéchs/seconde, toujours avec un filtre anti-repli, pour donner $x_2[n]$.

Concevez une chaîne purement numérique (rééchantillonnage et filtrage) qui fournit $x_2[n]$ à partir de $x_1[n]$

5.3 Chaîne sur-souséchantillonnage

Soit un signal $x_c(t)$ avec $X_C(j\Omega) = 0$ pour $|\Omega| \geq \frac{\pi}{T}$. Ce signal est échantillonné à $1/T$ pour donner $x[n] = x_c(nT)$. Ce signal est suréchantillonné par un facteur L, passe dans un filtre $H(e^{j\omega})$ et est ensuite sous-échantillonné par le même facteur L.

D'autre part, la réponse fréquentielle du filtre est donnée par :

$$H(e^{j\omega}) = \begin{cases} e^{-j\omega} & |\omega| < \dfrac{\pi}{L} \\ 0 & \dfrac{\pi}{L} < |\omega| \leq \pi \end{cases}$$

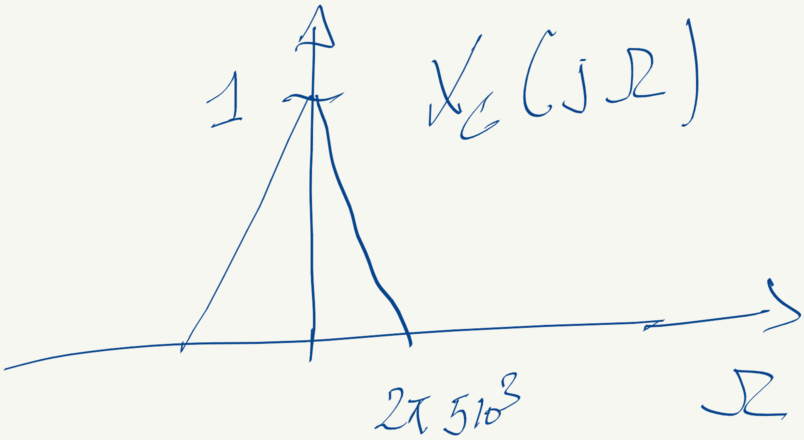
Que vaut $y[n]$?

5.4 Cascade de filtres

Soit un signal à temps discret $x[n]$, il passe dans un filtre passe-bas de réponse fréquentielle $H(e^{j\omega})$. La sortie de ce filtre est notée $v[n]$. Ce signal est suivi d'un sous-échantillonneur par M et donne $y[n]$. On a

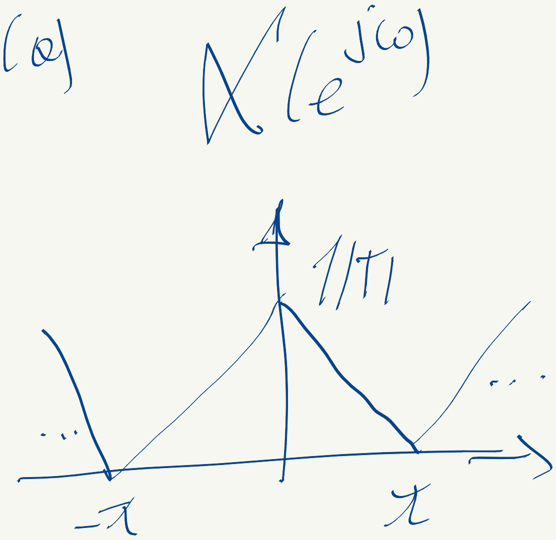
ΤΔ 5

3.1

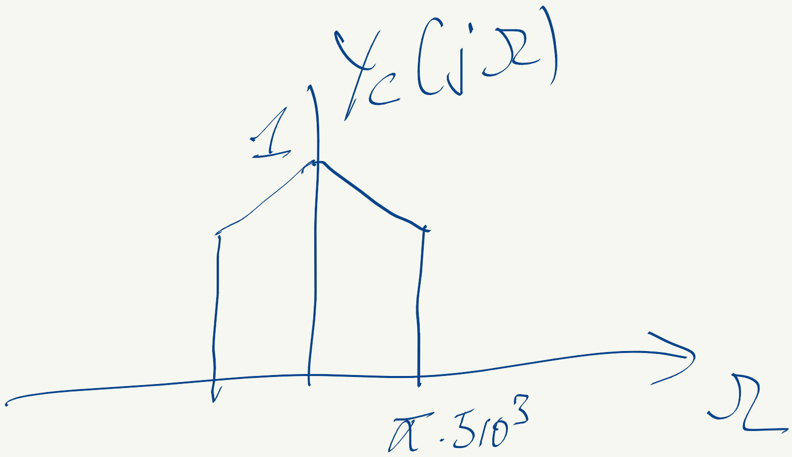


$X_c(j\Omega)$
1
$2\pi\, 5\, 10^3$
Ω

(a)

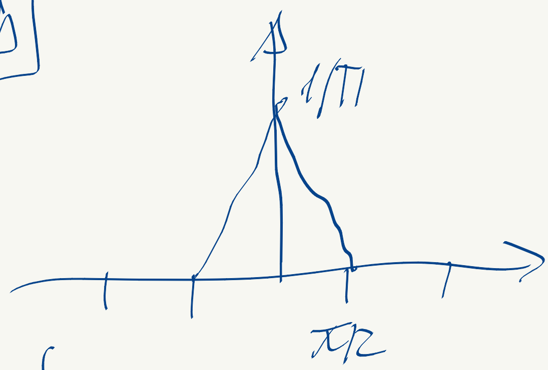


$X(e^{j\omega})$
$1/T_1$
...
$-\pi$
π

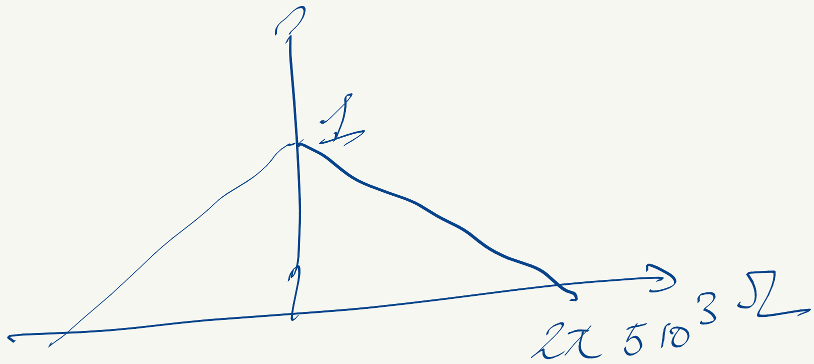


$Y_c(j\Omega)$
1
$\pi \cdot 5 10^3$
Ω

b)

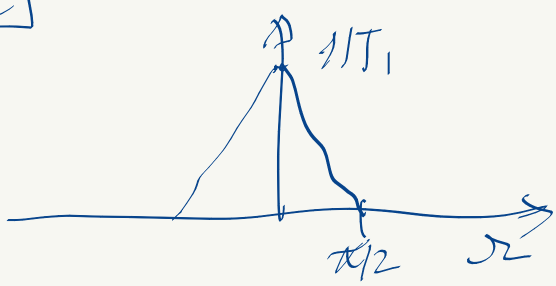


$1/T_1$
$\pi/2$

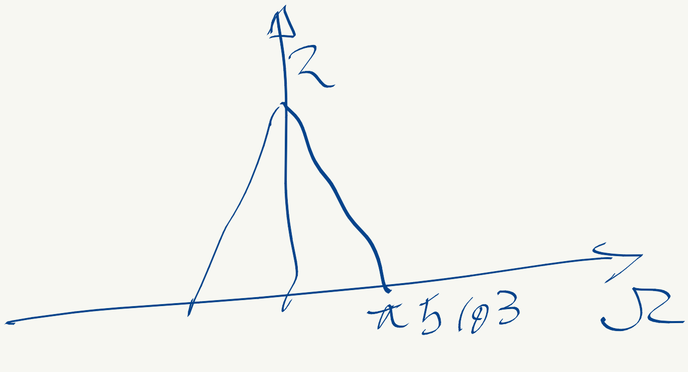


1
$2\pi\, 5\, 10^3$
Ω

c)



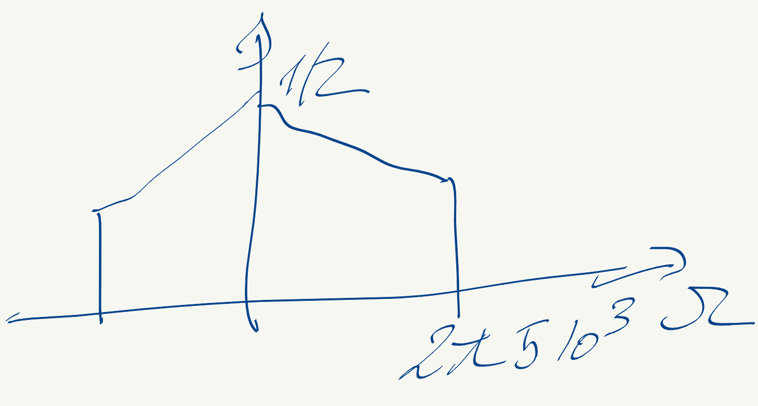
$1/T_1$
$\pi/2$
Ω



2
$\pi\, 5\, 10^3$
Ω

d)

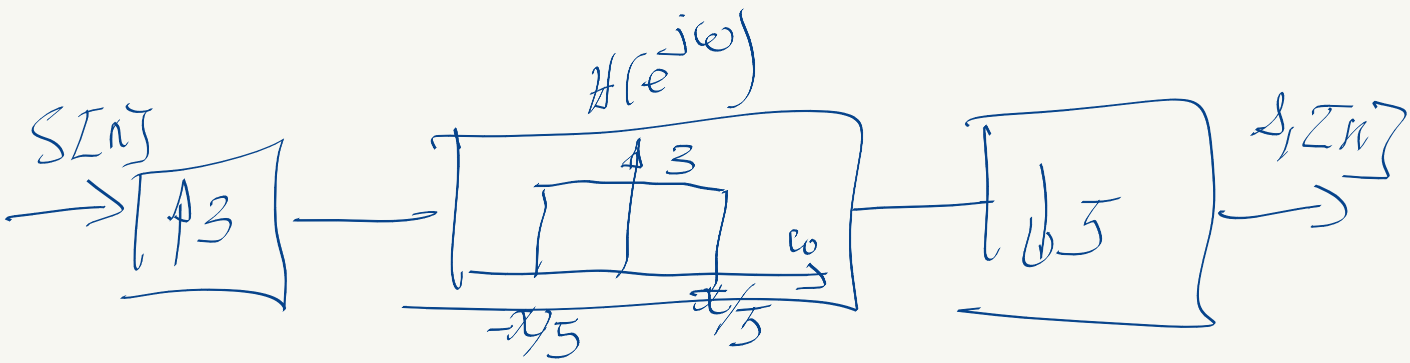
$1/T_1$
π



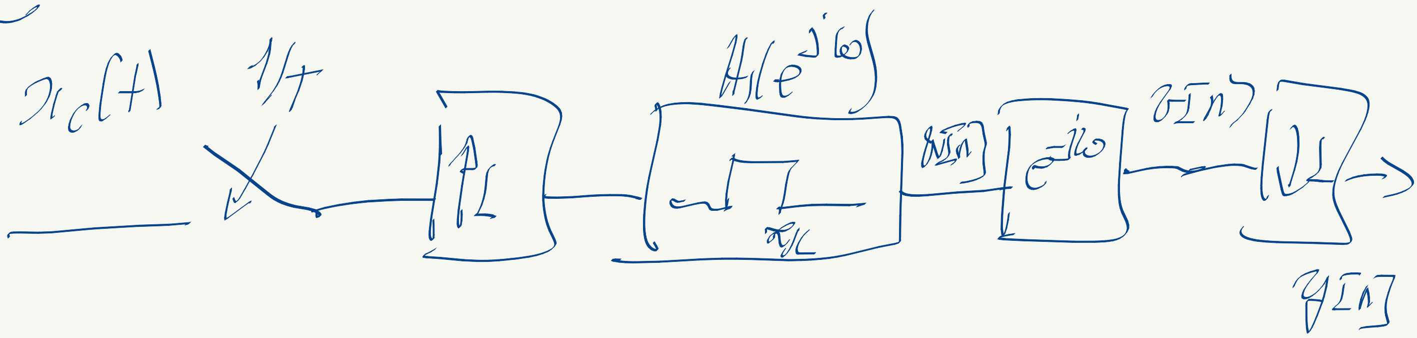
1/2
$2\pi\, 5\, 10^3$
Ω

5.2. note +: pas de repli de spectre car filtrage avant.

+ chgt de fréq. d'un rapport $\frac{6}{10} = \frac{3}{5}$



5.3



$x[n] = x_c(nT)$ car $x_c(t)$ à bande limitée (pas de repli)

Si pas de $e^{-j\omega}$

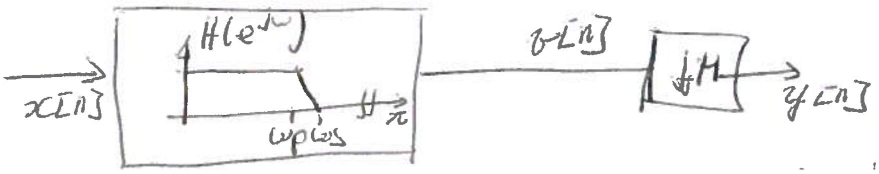
→ on retrouve le signal de départ avec un gain $\frac{1}{2}$ (le sur échantillonnage suivi du filtrage)

→ $y[n] = x_c(nT) = x[n]$

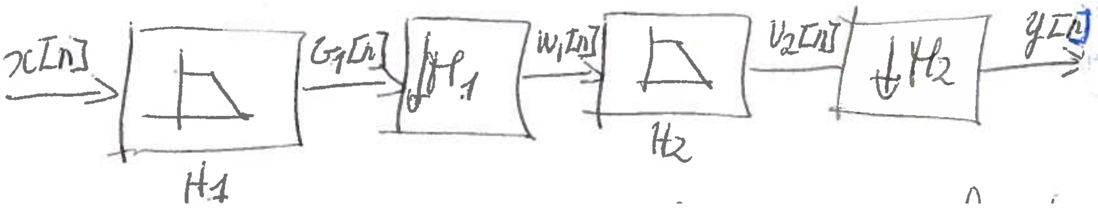
$e^{-j\omega} = z^{-1}$ à la fréquence $L f_e = \frac{L}{T}$ → décalage de T/L

⇒ $y[n] = x_c(nT - T/L)$

$$H(e^{j\omega}) = \begin{cases} 1 & |\omega| < \omega_s \\ x & \omega_p \leq |\omega| \leq \omega_s \\ 0 & \omega_s \leq |\omega| \leq \pi \end{cases}$$



De manière à relâcher les contraintes du filtre passe-bas, et donc la charge de calcul du filtre, on implémente cette chaîne par une cascade de filtres et de décimateurs (par exemple pour deux filtres, on a un filtre H_1 suivi d'un décimateur par M_1, suivi d'un filtre H_2, flanqué lui-même d'un décimateur par M_2.



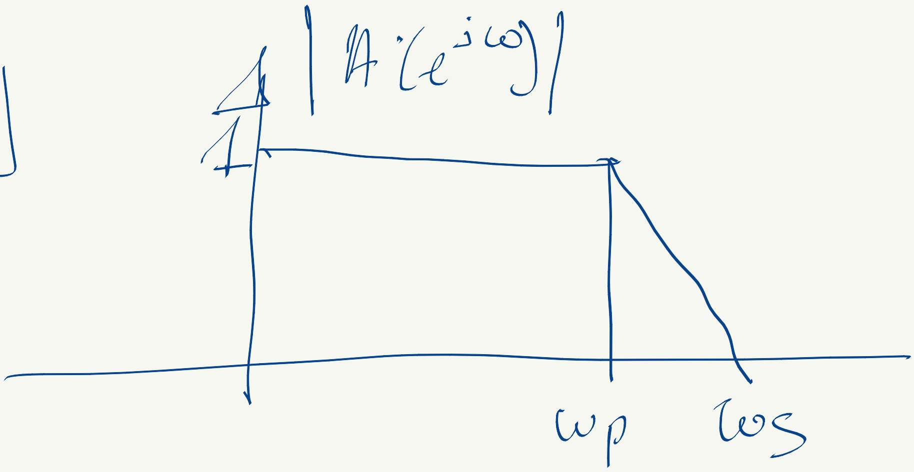
- Quelle est la valeur maximale de M en fonction de ω_s pour qu'il n'y ait pas de repli de spectre ?
- Soit $M = 100, \omega_s = \pi/100, \omega_p = 0.9\omega_s$. Si $x[n] = \delta[n]$ tracez le graphe de $V(e^{j\omega})$ et de $Y(e^{j\omega})$.
- Soit $M_1 = 50$ et M_2 = 2, avec $\omega_{p1} = 0.9\pi/100, \omega_{p2} = 0.9\pi/2, \omega_{s2} = \pi/2$, déterminez ω_{s1} tel que le filtre équivalent à la cascade ait la même fréquence de coupure que le filtre $H(e^{j\omega})$ (c'est-à-dire ω_s. Aidez-vous des graphes de $V_1(e^{j\omega}), W_1(e^{j\omega})$ et $V_2(e^{j\omega})$ si $x[n] = \delta[n]$.
- Soient les spécifications du filtre telle que l'ondulation dans la bande soit au maximum égale à 2 δ_p, avec δ_p = 0.01, et l'atténuation égale à $\delta_s = 0.001$ (soit 60 dB). Une approximation de l'ordre d'un filtre FIR passe-bas ayant ces spécifications est donnée par :

  $$N \simeq \frac{-10\log_{10}(\delta_p\delta_s) - 13}{2.3(\omega_s - \omega_p)} + 1$$

  Déterminez N pour l'implémentation à simple étage ainsi que le nombre d'opérations nécessaires en tenant compte de la symétrie du filtre.
- Si $x[n]$ est issu d'un échantillonnage à 50 Méch/sec, quel est la complexité de calcul nécessaire (en Mops/sec).
- En utilisant la valeur de ω_{s1} calculée plus haut, déterminez le nombre d'opérations nécessaires pour les filtres H_1 et H_2.
- L'oscillation dans la bande passante du filtre équivalent à la cascade pourrait être supérieure à δ_p, on utilisera alors, par exemple $\delta_{p1} = \delta_{p2} = 0.005$. Quelle sera l'influence sur N_1 et N_2 ?

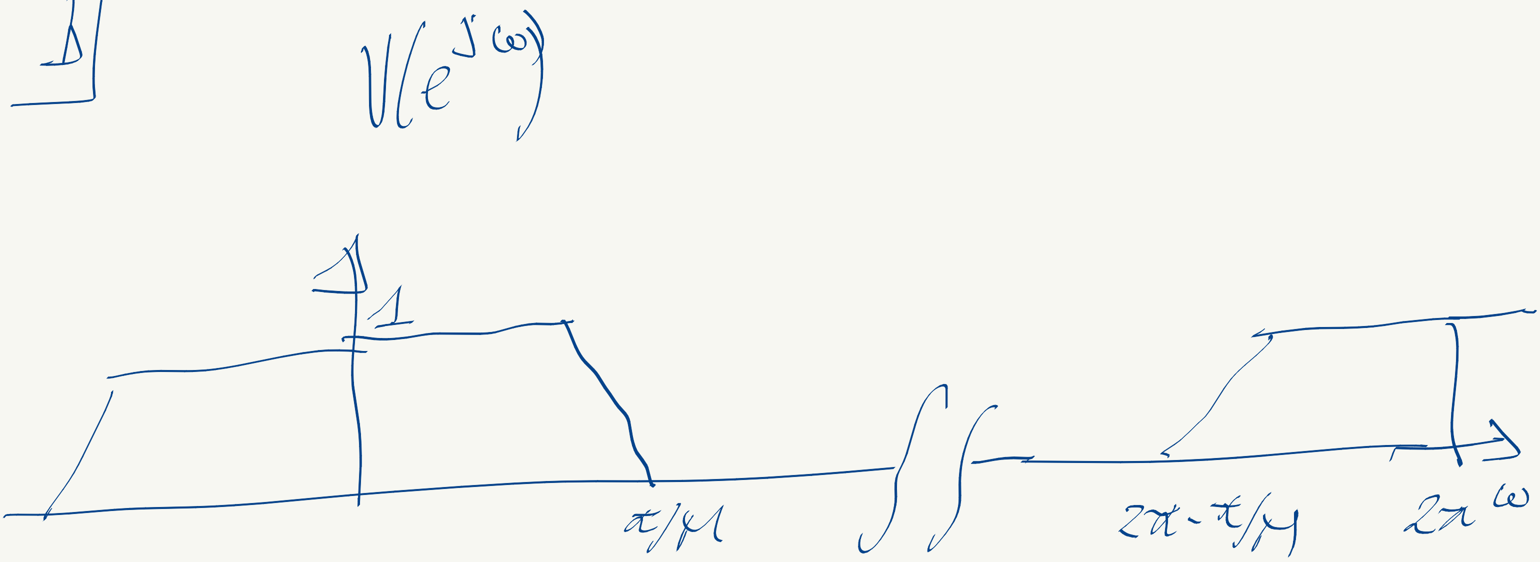
- Faut-il faire de même avec δ_{s1} et δ_{s2} ?
- Calculez la charge de calcul globale de l'implémentation en cascade.

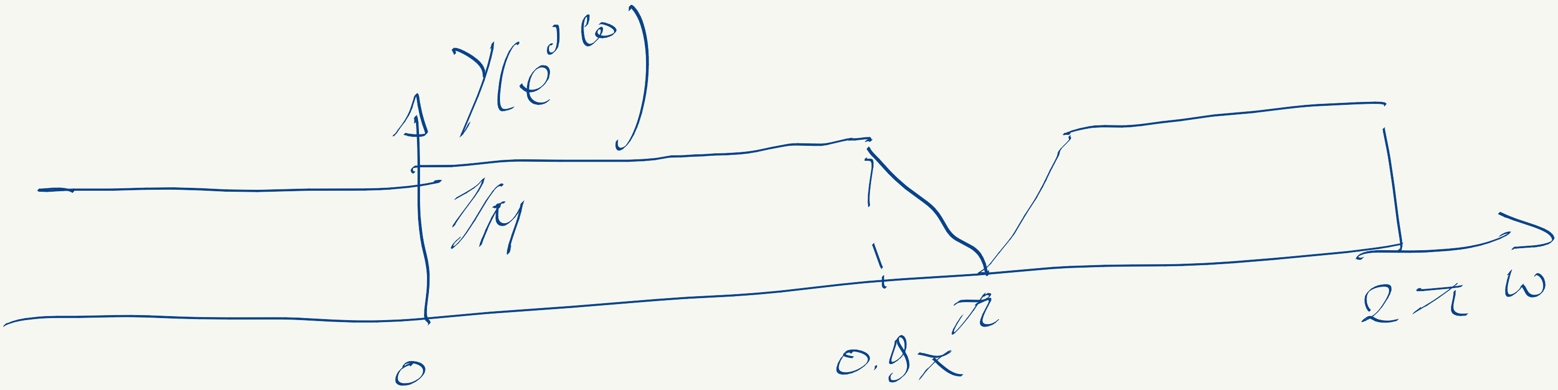
a)



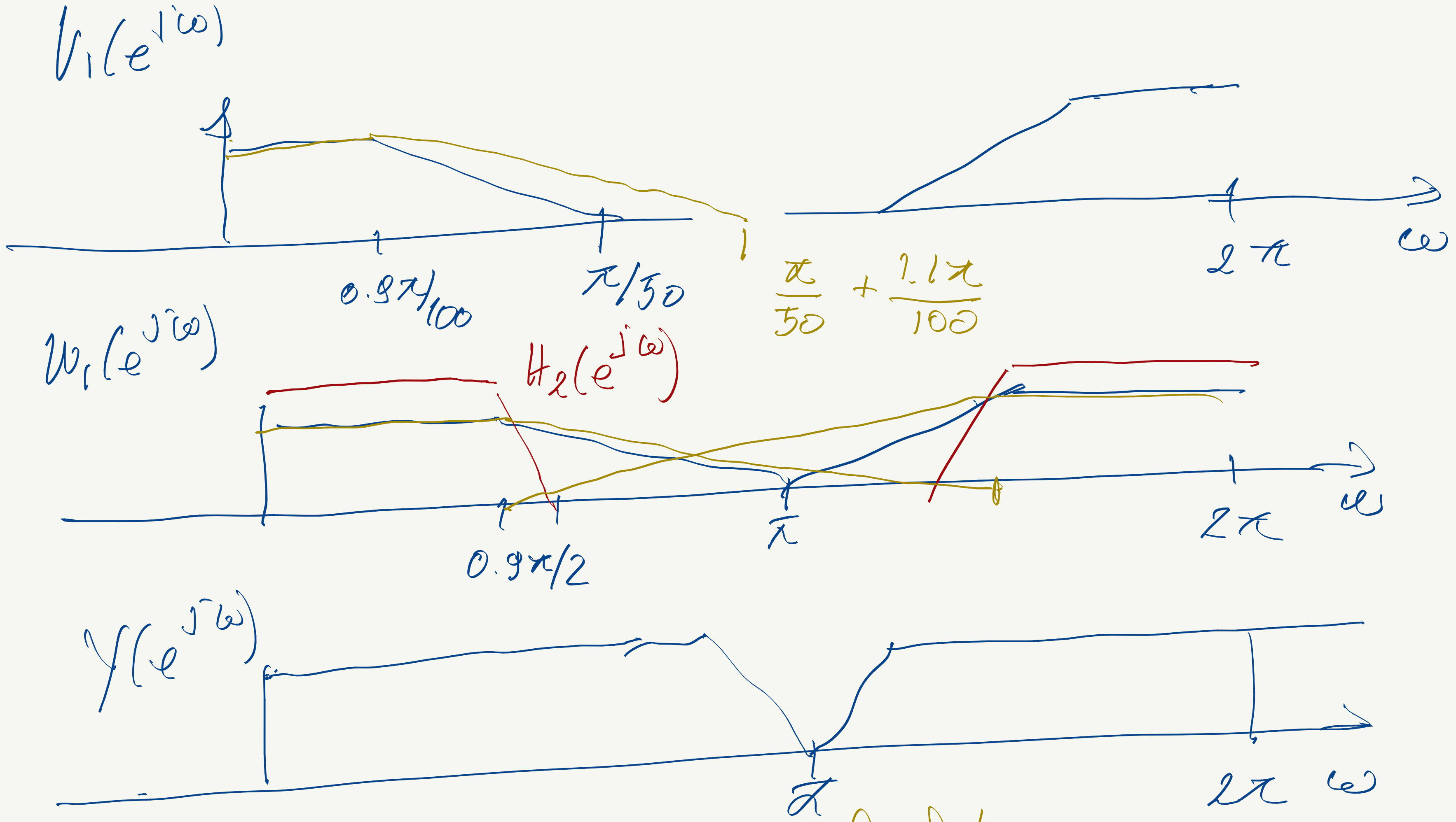
$\omega_s \le \frac{\pi}{M}$

b)





(1) en évitant le repli de spectre $\omega_{s1} = \pi/5$



Si on accepte le repli qui sera filtré par H_2

$$\omega_{s1} = \frac{\pi}{50} + 1.1\frac{\pi}{100} = \frac{3.2\pi}{100}$$

c)

d) pour $\delta p = 0{,}01$ $\delta s = 0{,}001$

$$\omega s - \omega p = \frac{0.1\pi}{100}$$

$$N = \frac{-10 \log_{10}(\delta p \delta s) - 13}{2{,}3(\omega s - \omega p)} + 1$$

$\simeq 5122$

pour $\omega s_1 - \omega p_1 = \frac{3.2\pi}{100}$ $\omega s_2 - \omega p_2 = \frac{\pi}{10}$

1er étage $N_1 \simeq 162$ 2eme étage $N_2 \simeq 53$

e) $Fe_1 = 50$ Mech/sec

cas 1 Cpxité $= Fe_1 \times \frac{N}{2}$

car FIR symetrique

$\simeq 128$ Gops/sec

cas 2 cpxité $= Fe_1 \cdot \frac{N_1}{2} + \frac{Fe_1}{50} \cdot \frac{N_2}{2}$

4.05 Gops + 26.5 Mops/sec $\simeq$ 4 Gops/sec

6 Filtres polyphases et Bancs de filtres

6.1 Filtre Polyphase : Implémentation

- Soit un filtre $H(z) = 1+2z^{-1}-3z^{-2}+2z^{-4}+z^{-5}$, donnez les filtres polyphases de type I et II pour M = 2 et pour $M = 4$.

  En Python, effectuer le filtrage d'une séquence de longueur 100 avec le filtre d'ordre 5 ci-dessus, par la méthode directe et en utilisant les filtres polyphases pour $M = 4$.

- Soit un décimateur d'ordre 4, implémentez une chaine "filtre anti-repli - décimateur d'ordre 4" avec des filtres polyphases. Vous partirez d'un filtre $H(z)$ FIR d'ordre 34 ou 35.

6.2 Filtre à reconstruction parfaite

- Implémentez le banc de filtres à reconstruction parfaite d'ordre vu au cours. Vous visualiserez les différents spectres (en entrée, en sortie et sur les points intermédiaires).

  Je vous suggère de créer les fonctions :

```
def divide(Y,Ho,H1):
    ....
    return UL, UH
```

```
def reconstruct(UL,UH,Go,G1):
    ....
    return Y
```

- Testez votre code avec les filtres à reconstruction parfaite d'ordre 33 et 99 donnés dans Moodle.

6.3 Banc de filtres d'analyse et de synthèse

- Fabriquez un banc de filtres à 4 branches, qui divise la bande passante en 4 parties égales, avec des filtres FIR de longueur 101. Effectuez l'analyse (figure 5 des transparents, partie gauche) et la synthèse (reconstruction, partie droite de la figure 5 (b) des transparents). Vérifiez tous les spectres.

- Implémentez le banc de filtres précédent à l'aide des filtres polyphase, avec la version faible complexité (figure 8(b) des transpartents). Vérifiez tous les spectres.

7 Homework Numéro 2 : multiplexage et démultiplexage par banc de filtres polyphases.

Vous êtes ingénieur audio, et vous recevez un multiplex audio formé de 24 canaux (analogiques). Chaque canal contien un signal codé en '.wav' à 44100 ksps, et donc de largeur 44100 Hz, et la fréquence centrale de chacune des bandes est de k 44100 kHz, où $k = [0..23]$. La largeur de bande totale est donc de 1036350 kHz. On vous demande de récupérer le canal x, en utilisant, outre les techniques de filtrage numérique, un banc de filtres.

Vous choisirez x comme étant le jour de votre naissance modulo 20 (ou modulo 20 +1), de telle sorte que le numéro de canal soit impair.

Dans un deuxième temps, vous remplacerez échangerez les contenus du canal x et du canal $y = 22$. Pour ce faire, vous utiliserez également un banc de filtres polyphases.

Vous rendrez un Notebook jupyter, comprenant les différents codes, avec :

- la démodulation par filtrage "simple" ;
- la démodulation par filtrage polyphase ;
- une instruction qui "joue" la musique du canal x ;
- le morceau de code qui démodule et "joue" la musique du canal 22 ;
- le code qui échange les canaux x et 22

A chaque étape, vous tracerez les réponses fréquentielles des filtres et les spectres des signaux obtenus.