TPs TNA – ESE – Cordi – Thebault

Chaîne de restitution sonore

ENSEA

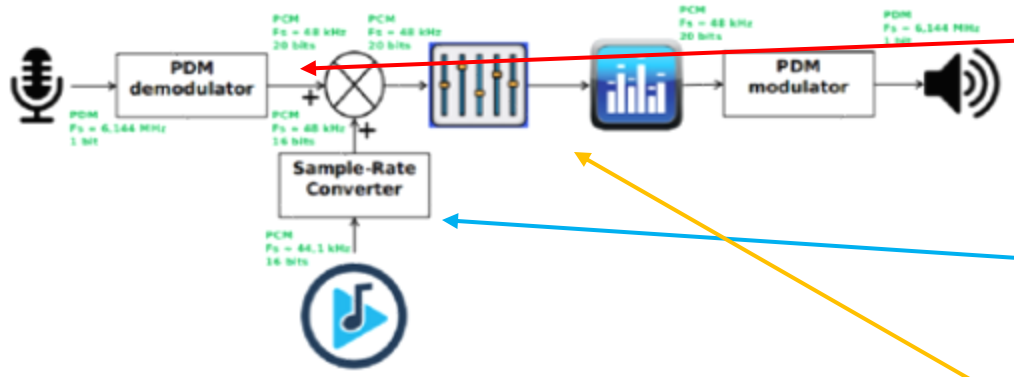
NELVEN THÉBAULT - HUGO CORDI

GITHUB (LIEN)

merci !

Contexte

Notre étude porte sur une chaîne de restitution sonore :



Elle est constituée :

- d'un microphone PDM qui restitue le signal code en PDM 1bit à 6,144MHz,
- d'un player audio qui émet un signal audio provenant d'un CD (44.1 kHz - 16 bits),
- d'un égaliseur graphique 10 bandes en octaves,
- d'un vu-mètre 10 bandes en octave,
- d'un modulateur PDM à 6,144MHz.

- 1ère séance

  PDM demodulator - PDM modulator

- 2ème séance

  Sample-Rate Converter (SRC)

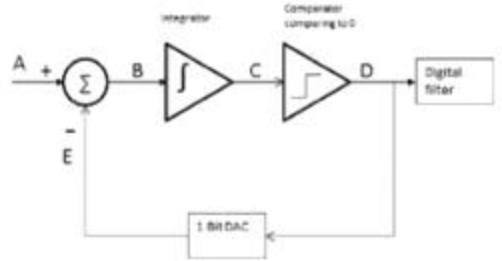
- 3ème séance

  Banc de filtres

lère séance → PDM demodulator - PDM modulator

Séance 1: PDM demodulator - PDM modulator.

PDM_modulator



Le schéma ci-dessus présente un synoptique d'un modulateur PDM. Expliquez le fonctionnement d'un modulateur PDM.

PDM demodulator

Le signal source proposé *pdm_in.mat* contient un signal audio. Il serait intéressant de l'observer dans les domaines temporel et fréquentiel. Pour vous convaincre définitivement de son caractère audio et si vous l'écoutiez ?

Proposer des méthodes de démodulation du signal PDM afin d'obtenir le signal avec les caractéristiques décrites sur le synoptique principal (PCM - 48kHz - 20 bits). Préciser ce qu'impliquent ces caractéristiques. Proposez plusieurs alternatives prenant en compte la linéarité de la phase, les délais de propagation de groupe, etc.

- Objectif

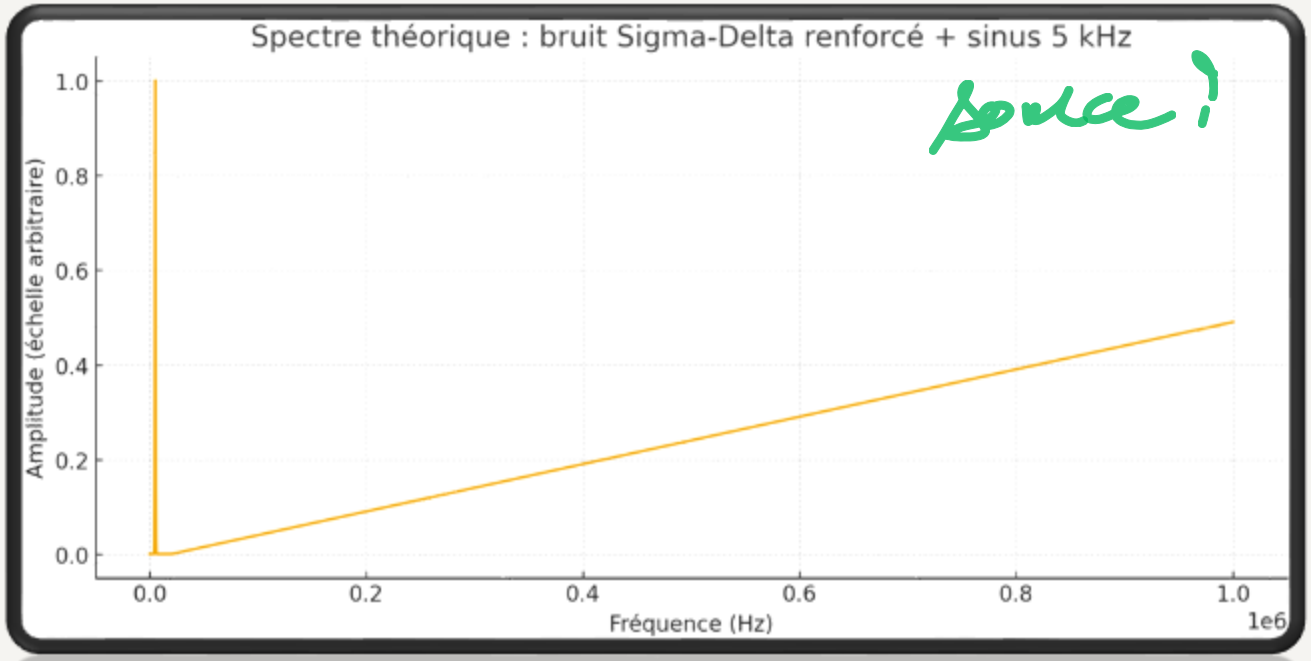
Récupérer le signal audio en PDM – 6,144 MHz – 1 bit et designer le démodulateur pour récupérer le signal en PCM – 48 kHz – 20 bits.

On va donc faire un décimateur :

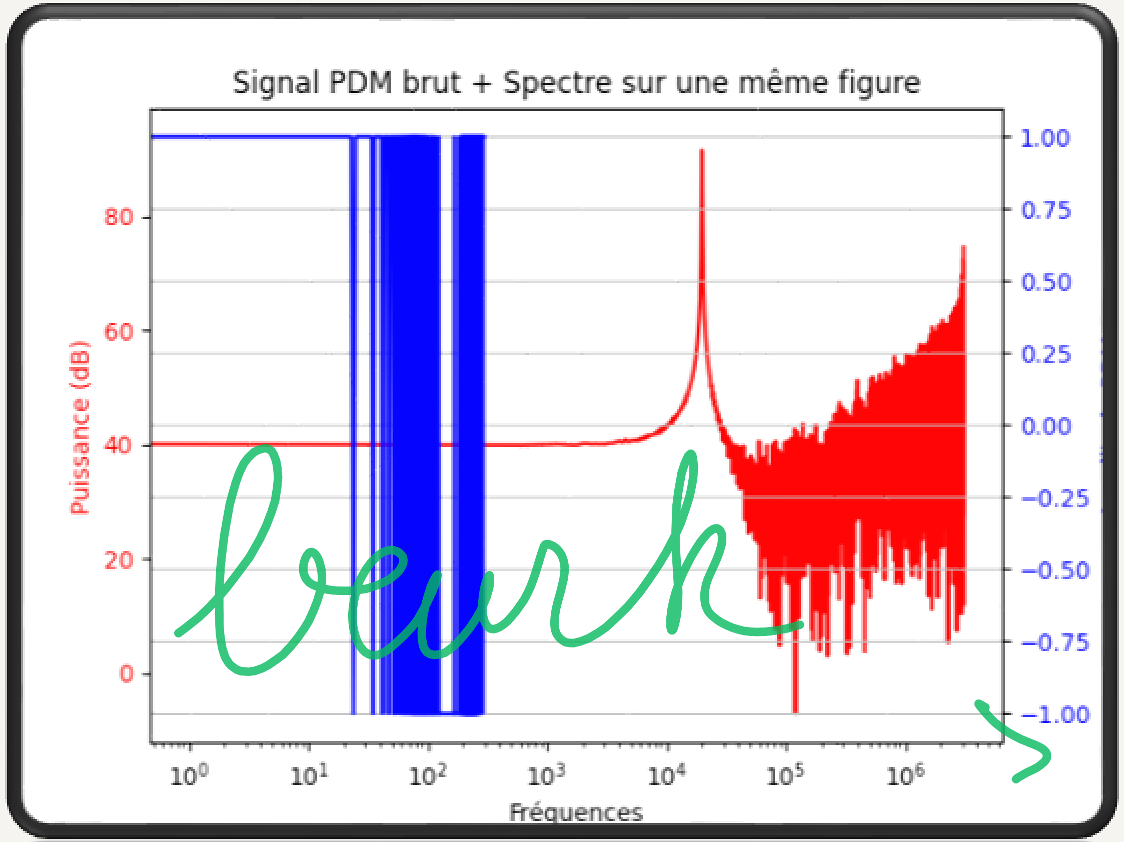
- Filtrage
- Down sampling (facteur de ↓128 → 6144/48)

Modulateurs PDM

- Le microphone utilise un ADC **sigma-delta (Σ-Δ)** pour convertir le signal analogique de la voix en un signal numérique normalisé dans l'intervalle **[-1 ; 1]**. Ce type d'ADC est très répandu car il offre une **excellente précision**.
  Il présente une **caractéristique notable** : un **bruit très faible** dans la **bande passante utile**, puis un **bruit qui augmente linéairement avec la fréquence**. Cela ne pose pas de problème puisqu'un filtre passe-bas suit immédiatement la conversion.



Observation du signal en entrée



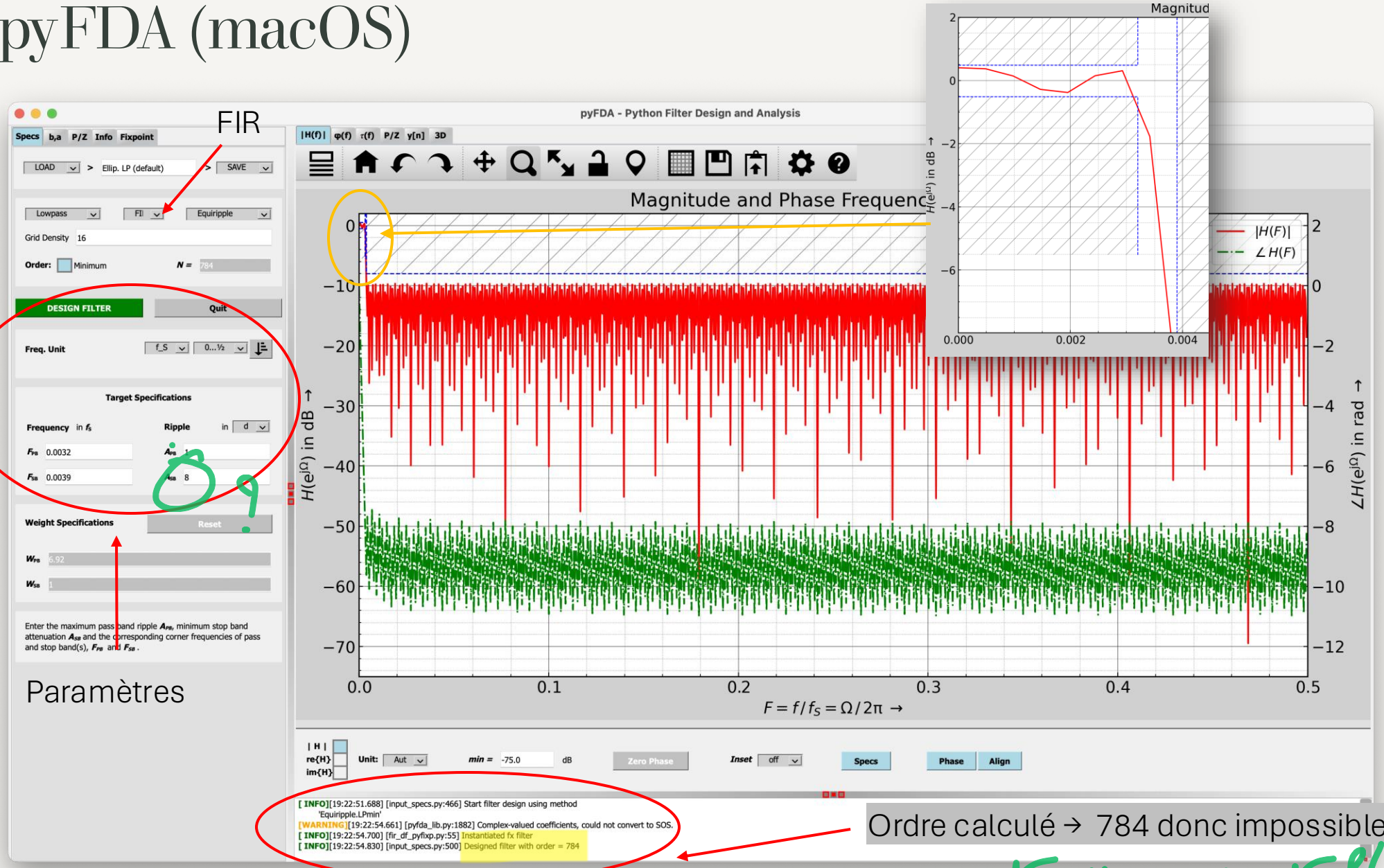
- En bleu le signal PDM
- En rouge son spectre qui comporte une bande utile de 20 à 20 kHz (symétrie jusqu'à 40 kHz)
  On observe bien un bruit jusqu'à 6,144 MHz qui est linéaire par rapport à la fréquence et que l'on va devoir filtrer

Observation du signal en entrée

On design donc via pyFDA plusieurs filtres avec les caractéristiques suivantes :

- f_c = 24 kHz (bande utile) mais en sortie on veut du $f_{s,out}$ = 48 kHz donc fs_{in}/fs_{out}= 6144/48 = 128. Donc stopband à $f_{s,out}/2 = f_c$ = 24 kHz donc on a en non normalisé :
  - f_p =20 kHz, f_{sb} = 24 kHz et donc Δf = 4 kHz
  - En normalisé sur F = f/f_s [0;0,5] on obtient : F_p = 20 / 6144 = 0,0032 et F_{sb} = 0,0039
- Atténuation ? On veut du bruit inférieur au quantum soit $2/2^{20} = 2^{-19}$ → $20\log(2^{-19})$=-114 dB soit environ 120 dB comme ça on est bien sur 20 bits → Faux, il faut intégré ce qu'il y a après 20 kHz et avant comme ça avec le rapport on obtient le SNR
  - On peut donc récupérer les dB d'atténuations via la formule 6,02N+1,76 = 8 dB car N = 1 bit pour A_{SB} et on choisit d'autoriser un ripple de A_{PB} = 1 dB dans la bande passante.
- On obtient alors un SNR_{dB} = 1,12 dB avant filtrage.

pyFDA (macOS)



Ordre calculé → 784 donc impossible à réaliser

ça peut passer suivant l'application

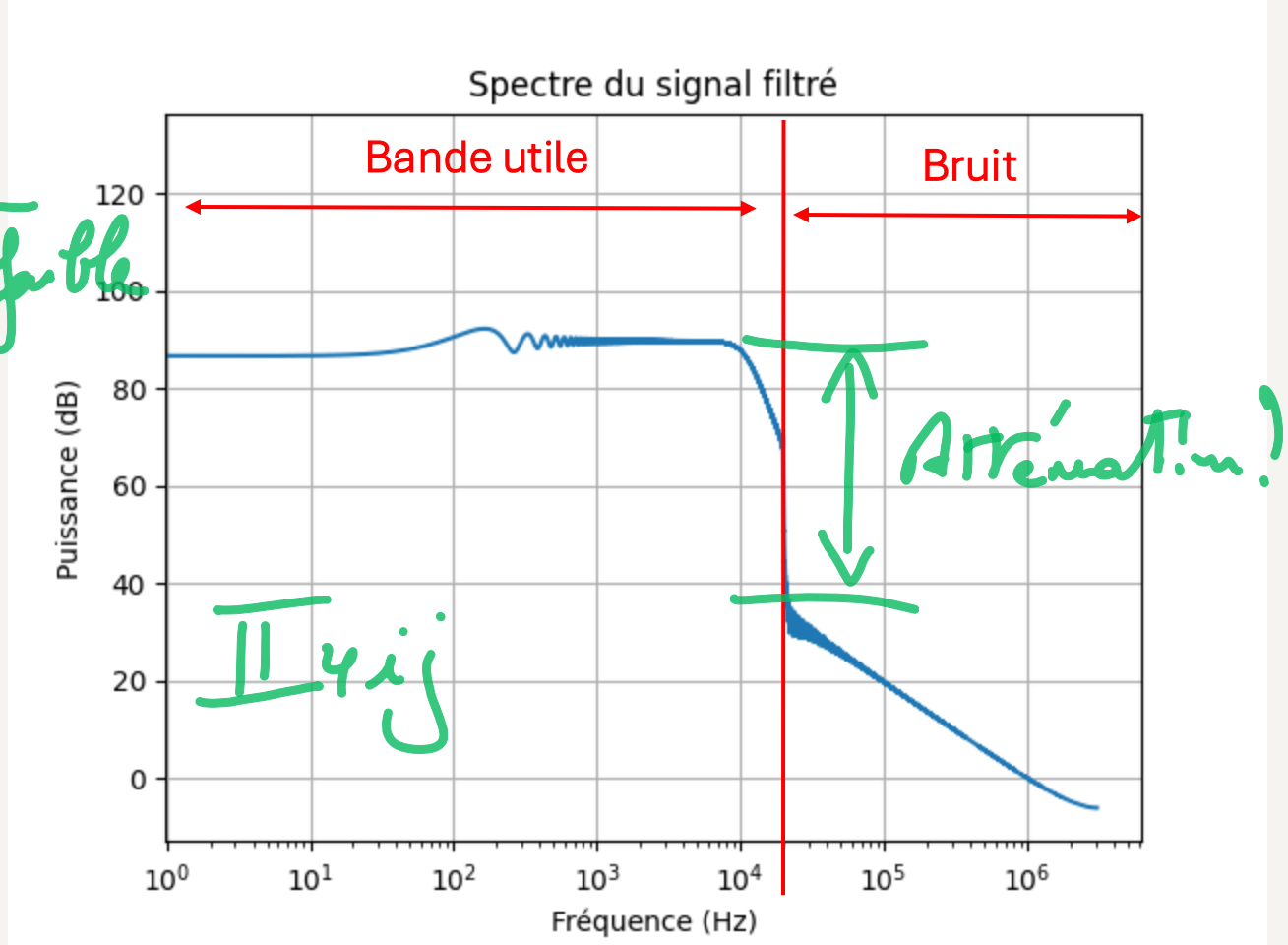
Côté Python : choix du filtre

Nous avons choisi d'utiliser un filtre IIR Butterworth suivi d'une décimation, principalement pour son faible coût calculatoire et sa faible latence, essentiels dans les applications temps réel comme l'audio live ou la voix où le délai de traitement doit être minimal.

Bien que la phase non linéaire de l'IIR puisse introduire une légère distorsion des transitoires, cette méthode offre un bon compromis entre performance et complexité, évitant l'usage coûteux en ressources des filtres FIR à phase linéaire. Ce choix est justifié lorsque la priorité est la réactivité du système plutôt que la perfection audiophile, ce qui correspond parfaitement à nos contraintes opérationnelles.

Côté Python, filtrage

- Chargement du signal PDM depuis un fichier .mat
- Analyse spectrale initiale du signal brut avec calcul du SNR avant/après 20 kHz
- Filtrage passe-bas IIR Butterworth d'ordre 4 avec fréquence de coupure adaptée au facteur OSR
- Décimation par OSR, normalisation et quantification en 20 bits pour obtenir le signal PCM
- Analyse spectrale du signal filtré et affichage des coefficients du filtre Butterworth (b, a)



Côté Python, Résultats

- SNR avant filtrage : 1.12dB
- SNR après filtrage : 51.21 dB
- Coefficients du filtre de Butterworth :

a =[1. 0, -3.97113898, 5.91383269, -3.91424494, 0.97155124]

b = [9.16727374e-10, 3.66690950e-09, 5.50036425e-09, 3.66690950e-09, 9.16727374e-10]

- Partie non comprise : la notion de niveau de bruit cible. Nous ne savons pas encore comment relier le SNR obtenu au fait que le bruit doit rester inférieur à un quantum (1 LSB).

à rediscuter ensemble lors de la prochaine séance