UNIVERSITÉ DE SHERBROOKE

Faculté de génie

Département de génie électrique et génie informatique

**Plan de Validation APP6**

Traitement du signal sur µC

APP6

Présenté à

Jean-Baptiste Michaud

Présenté par

Mathieu Desautels – DESM1210

Félix Boivin – BOIF1302

Université de Sherbrooke – 8 avril 2024

Table des matières

[A1 - Calcul de la résolution fréquentielle du spectre 4](#_Toc163392305)

[Description : 4](#_Toc163392306)

[Résultat attendu : 4](#_Toc163392307)

[Test unitaire à exécuter : 4](#_Toc163392308)

[Résultat obtenu : 4](#_Toc163392309)

[A2 & A3 – Calcul du spectre d’entrée, effet du fenêtrage 4](#_Toc163392310)

[Description : 4](#_Toc163392311)

[Simulations : 4](#_Toc163392312)

[Test unitaire à exécuter : 5](#_Toc163392313)

[Résultat obtenu : 5](#_Toc163392314)

[A4 – Calcul de l’indice de la fréquence à amplitude maximale 6](#_Toc163392315)

[Description : 6](#_Toc163392316)

[Résultat attendu, conditions du test : 7](#_Toc163392317)

[Test unitaire à exécuter : 7](#_Toc163392318)

[Résultat obtenu : 7](#_Toc163392319)

[B0 & B1 – Calcul de la FFT du signal d’entrée rallongé à 4N (méthode overlap & save) 7](#_Toc163392320)

[Description : 7](#_Toc163392321)

[Résultats attendus, conditions du test : 7](#_Toc163392322)

[Test unitaire à exécuter : 8](#_Toc163392323)

[Résultat obtenu : 8](#_Toc163392324)

[B2 – Filtrage dans le domaine fréquentiel 8](#_Toc163392325)

[Description : 8](#_Toc163392326)

[Résultats attendus, conditions du test : 9](#_Toc163392327)

[Test unitaire à exécuter : 9](#_Toc163392328)

[Résultats obtenus : 9](#_Toc163392329)

[B3 & B4 – FFT inverse, extraction de la trame filtrée 10](#_Toc163392330)

[Description : 10](#_Toc163392331)

[Résultats attendus, conditions du test : 10](#_Toc163392332)

[Test unitaire à exécuter : 10](#_Toc163392333)

[Résultats obtenus : 10](#_Toc163392334)

[C1 – Filtres IIR 12](#_Toc163392335)

[Description : 12](#_Toc163392336)

[Résultats attendus, conditions du test : 13](#_Toc163392337)

[Test unitaire à exécuter : 13](#_Toc163392338)

[Résultat obtenu : 13](#_Toc163392339)

[ANNEXE A : GRAPHIQUE PYTHON FENÊTRAGE 16](#_Toc163392340)

[ANNEXE B : GRAPHIQUE MPLAB FENÊTRAGE 17](#_Toc163392341)

[ANNEXE C : RÉPONSE EN FRÉQUENCES DES FILTRES FIR 18](#_Toc163392342)

[ANNEXE D : GRAPHIQUES PYTHON DU FILTRE IIR 19](#_Toc163392343)

[ANNEXE E : FONCTION DE TRANSFERT FILTRE IIR 20](#_Toc163392344)

Tables des illustrations :

[Figure 1 : windows.h python 6](#_Toc163392345)

[Figure 2 : windows.h MPLAB 6](#_Toc163392346)

[Figure 3 : maxAmplFreq valeurs sur MPLAB 7](#_Toc163392347)

[Figure 4 : FFT d'entrée en utilisant overlap & save 8](#_Toc163392348)

[Figure 5 : FFT Python sans utiliser la méthode overlap & save 8](#_Toc163392349)

[Figure 6 : Réponse fréquentielle des filtres H3 à H7 par python 9](#_Toc163392350)

[Figure 7 : Sinus 300 Hz dans filtre H7 11](#_Toc163392351)

[Figure 8 : Sinus 500 Hz dans filtre H7 11](#_Toc163392352)

[Figure 9 : Sinus 700 Hz dans filtre H7 11](#_Toc163392353)

[Figure 10 : Filtre passe-bas H7 11](#_Toc163392354)

[Figure 11 : Sortie du filtre IIR à 900 Hz 14](#_Toc163392355)

[Figure 12 : Sortie du filtre IIR à 1000 Hz 14](#_Toc163392356)

[Figure 13 : Sortie du filtre IIR à 1100 Hz 15](#_Toc163392357)

[Figure 14 : Résultat Python filtre IIR Q2.13 15](#_Toc163392358)

[Figure 15 : Fichier filterFIRcoeffs.h dans MPLAB filtre IIR 15](#_Toc163392359)

[Figure 16 : Graphique Python du spectre de fréquence sur une sinus de 2000 Hz 16](#_Toc163392360)

[Figure 17 : Graphique DMCI du spectre de fréquence sur une sinus de 2000 Hz 17](#_Toc163392361)

[Figure 18 : Graphiques Python des réponses en fréquences des filtres FIR 18](#_Toc163392362)

[Figure 19 : Graphiques des test python du filtre IIR 19](#_Toc163392363)

[Figure 20 : Fonction de transfert du filtre IIR 20](#_Toc163392364)

# A1 - Calcul de la résolution fréquentielle du spectre

## Description :

Calcul de la valeur de la variable spectralResolution (type double). Cette variable est utilisée pour savoir quelle sera la variation dans le système. Une fréquence d’échantillonnage haute remplira les variables très rapidement sans nécessairement finir un cycle de sinus complet, une fréquence haute aura donc une résolution plus basse. C’est la résolution de fréquence, cela représente le maximum de précision possible du système.

## Résultat attendu :

## Test unitaire à exécuter :

Afficher sur le LCD la valeur obtenue en divisant la fréquence d’échantillonnage par le nombre de coefficients qui correspond à la longueur de l’array de la FFT. La valeur de la fréquence d’échantillonnage est indiquée lors de l’allumage du système sur le LCD. La nombre d’échantillons (*N*) est défini dans la variable *FFT\_LEN,*  un calcul peut être fait sur calculatrice pour valider que la valeur sur l’écran (après l’appui de BTND) est bien la même.

## Résultat obtenu :

Une image contenant texte, Appareils électroniques, Ingénierie électronique, câble

Description générée automatiquement

# A2 & A3 – Calcul du spectre d’entrée, effet du fenêtrage

## Description :

En premier lieu, la transformée de Fourrier du signal d’entrée est calculée avec la fonction mips\_fft32 et ensuite le module carré du spectre est calculé en dB. Le résultat est stocké dans le tableau debugBuffer1 et est affiché par les graphiques de DMCI. Ces étapes sont répétées pour chaque type de fenêtrage utilisé, en occurrence le fenêtrage rectangulaire et le fenêtrage Hanning.

Les coefficients sont calculés grâce à un code python et stocker dans un fichier intitulé windows.h. Le fenêtrage rectangulaire sont des valeurs de 8192 (1 x 213) et le fenêtrage de hanning sont des valeurs variantes eux aussi sur une base 213.

## Simulations :

**Graphique Python signal test 2 kHz :**

À la Figure 16, il est possible de voir trois figures. Dans la première figure il est possible de voir le signal du sinus sur un échelle de temps. Ensuite, dans la deuxième figure il est possible de voir les spectres de l’onde sur une échelle de 1024 (longueur de *FFT\_LEN*). Finalement, dans la figure 3 il est possible de voir les mêmes spectres que la figure 2, mais sur une échelle de fréquence logarithmique.

**Graphiques DMCI :**

À la Figure 17, il est possible de voir 4 graphiques. Le graphique intitulé Channel 1 est un graphique identique à celui montré au Channel 3, uniquement Channel 1 est montré sur seulement 65 points au lieu de 1024. Par la suite, le Channel 2 est la réponse d’un signal de 2000 Hz en entrée, celle-ci avec un fenêtrage rectangulaire. Finalement, le Channel 4 est le même que le Channel 2, cependant celui-ci est équipé d’un filtre hanning.

## Test unitaire à exécuter :

**Test fenêtre rectangulaire :**

Afficher les graphiques de la FFT d’un sinus de 2000Hz avec un fenêtrage rectangulaire sur python. Afficher le même graphique sur MPLAB grâce au DebugBuffer1 en appliquant la fenêtre windows.h fourni de base avec les fichiers de départ sur une onde de 2000 Hz pur. Les pics devraient être placer au même endroit sur une base de 1024 (*FFT\_LEN*) soit environ 102 et 922. Les amplitudes des pics ne sont pas importantes en soit même la variation entre le bruit et le pic est l’important soit environ 50 dB

**Test fenêtre Hanning :**

Afficher les graphiques de la FFT d’un sinus de 2000Hz avec un fenêtrage hanning sur python. Afficher le même graphique sur MPLAB grâce au DebugBuffer1 en appliquant la fenêtre sur une base Q2.13 calculé grâce à python sur une onde de 2000 Hz pur. Les pics devraient être placés au même endroit sur une base de 1024 (*FFT\_LEN*) soit environ 102 et 922. Les amplitudes des pics ne sont pas importantes en soit même la variation entre le bruit et le pic est l’important soit environ 100 dB

**Test du fichier windows.h :**

Il faut assurer que les valeurs de pythons sont les même que les valeur afficher dans le fichier windows.h.

## Résultat obtenu :

**Test fenêtre rectangulaire :**

Le deuxième graphique de la Figure 16 montre le résultat de la FFT avec l’axe des x représentant les fréquences normalisées sur le nombre d’échantillons N = 1024. En comparant la courbe bleue, soit celle correspondant au fenêtrage rectangulaire, au channel 2 de la Figure 17, il est possible de remarquer que la forme est très similaire. En effet, les pics de la fréquence positive et de celle négative sont positionnés respectivement à k = 102 et k = 922. De plus, le delta de gain en dB entre le point maximum et minimum est d’environ 60 dB dans les deux graphiques.

**Test fenêtre Hanning :**

Le deuxième graphique de la Figure 16 montre le résultat de la FFT avec l’axe des x représentant les fréquences normalisées sur 1024. En comparant la courbe orange, soit celle correspondant au fenêtrage Hanning, au channel 4 de la Figure 17, il est possible de remarquer que la forme est très similaire. Comme avec le fenêtrage rectangulaire, les deux pics de fréquences sont aux positions k = 102 et k = 922. Le delta de gain entre le maximum et le bruit est beaucoup plus gros qu’avec le fenêtrage précédent. Avec python, le delta est d’environ 175 dB, mais avec le DMCI, le delta est d’environ 80 dB.

**Test du fichier windows.h :**

Il est possible de voir que les valeurs du fenêtrage hanning Q2.13 de python (Figure 1) est retrouver parfaitement dans le fichier windows.h de MPLAB (Figure 2).

****

Figure 1 : windows.h python

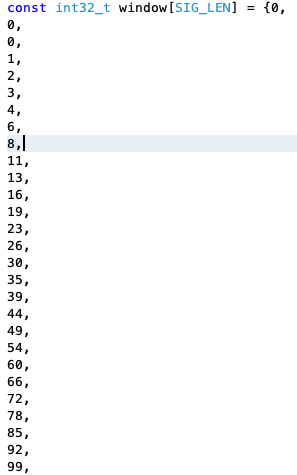
****

Figure 2 : windows.h MPLAB

# A4 – Calcul de l’indice de la fréquence à amplitude maximale

## Description :

Calcul de la valeur de la variable maxAmplFreq (type int)

Cette fonction permet de simplement calculer la fréquence qui a le plus grand gain. MaxAmplFreq est la fréquence en Hz qui a le gain le plus élevé, maxN est l’indice k qui correspond à la fréquence normalisée sur 1024 et spectralResolution est ce qui a été vu au point A1.

## Résultat attendu, conditions du test :

Lorsqu’un sinus de 2 kHz est à l’entrée, la fréquence positive avec un plus grand gain est à l’indice k = 102. La résolution spectrale est de 19,53 Hz. Il est donc possible de calculer :

C’est normal que maxAmplFreq ne soit pas exactement à 2 kHz, puisque la résolution spectrale est relativement grande et ne permet donc pas une très bonne précision.

## Test unitaire à exécuter :

Pour le test de l’amplitude maximale, un break point doit être fait dans le code avec un watch sur la variable maxN et maxAmplFreq.

## Résultat obtenu :



Figure 3 : maxAmplFreq valeurs sur MPLAB

# B0 & B1 – Calcul de la FFT du signal d’entrée rallongé à 4N (méthode overlap & save)

## Description :

Le signal d’entrée est décomposé en deux buffers soit currentInBuffer et previousInBuffer. CurrentInBuffer est celui dans lequel le programme écrit, une fois qu’il est plein le currentInBuffer devient le previousInBuffer et ainsi de suite. Donc pour faire un overlap and save, il suffit d’utiliser tous les blocs de previousInBuffer et le dernier bloc de currentInBuffer. Étant donné les deux buffers sont de 768 (3N) et la FFT nécessite un array de 1024 (4N), les 256 (1N) derniers bits de currentInBuffer seront utilisés.

Ce signal de 4N est placer dans inFFT dans la section réelle uniquement pour commencer, puisque le signal ne contient pas de nombre imaginaire. InFFT est utilisé pour le calcul de la FFT en utilisant mips\_fft32. Cette sortie est d’une longueur de 4N et elle contient des nombres réels et des nombres imaginaires. La sortie doit être multipliée avec les coefficients des filtres, et puisqu’il y a des chiffres imaginaires et réels, il est nécessaire d’utiliser le produit des nombres complexe.

## Résultats attendus, conditions du test :

Si le overlap and save est bien réalisé, l’onde de 2 kHz devrait rester de forme sinusoïdale et le spectre fréquentielle de cette onde devrait présenter des pics clairs à k=102 et k=922 (les indices k correspondants aux fréquences 2 kHz et 18 kHz). Si le overlap and save est mal fait, des fréquences autres que celles-ci seront visible sur le spectre puisque le signal ne sera plus sinusoïdal.

## Test unitaire à exécuter :

Calculer la FFT d’une onde de 2 kHz allongée avec la méthode overlap and save et comparer la sortie du debugBuffer1 du DMCI avec le spectre de fréquence théorique créé par python. Le script python calcul sans utiliser la méthode overlap and save, donc si les deux sont identiques cela confirme que overlap and save fonctionne.

## Résultat obtenu :

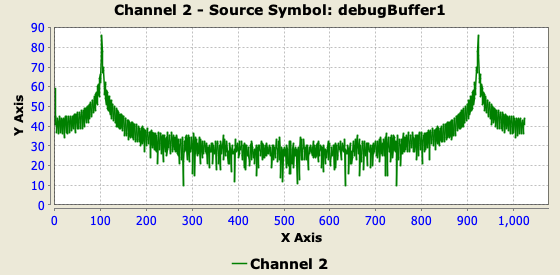


Figure 4 : FFT d'entrée en utilisant overlap & save

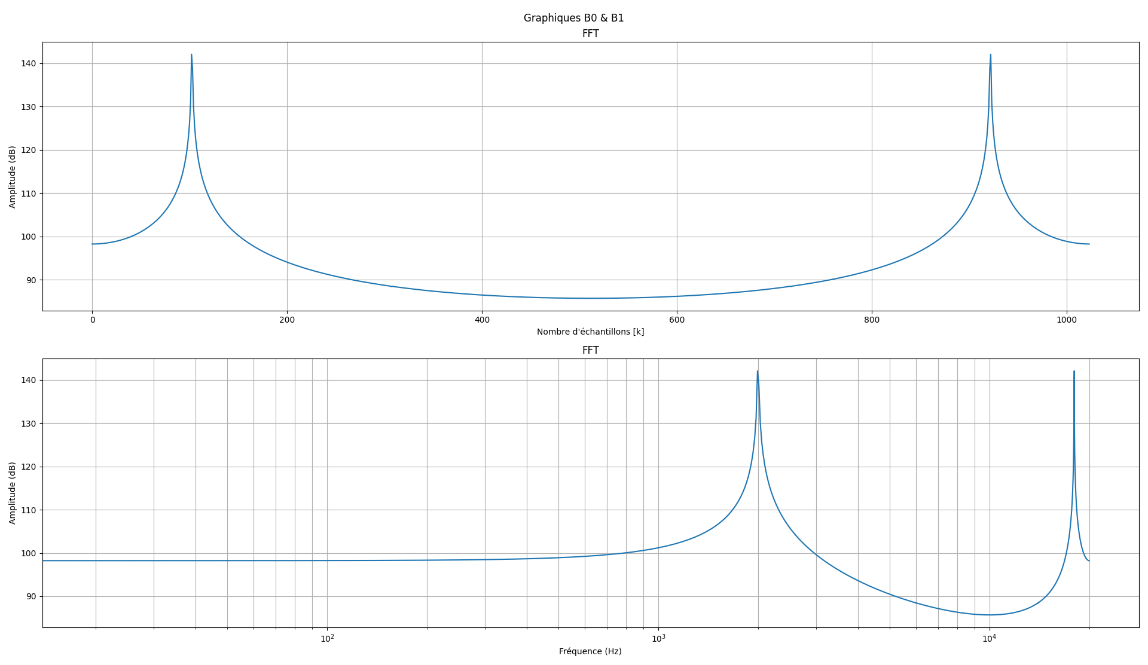


Figure 5 : FFT Python sans utiliser la méthode overlap & save

# B2 – Filtrage dans le domaine fréquentiel

## Description :

**Pycharm :**

La première étape est la création des filtres avec python. Les coefficients en sortie des fonctions sont en float, mais il faut les convertir en Q2.13. Pour ce faire, on multiplie chaque coefficient par 213. Ensuite, on converti les données en int pour seulement garder la partie des entiers. En affichant les données, on devrait avoir seulement des entiers entre -32 768 et 32 767 (Min et max de 16 bits en C2).

**MPLab :**

Rendu dans MPLab, il faut faire la FFT en Q15 du signal d’entrée de 2kHz. La fonction mips\_fft32 sort un résultat du même type que son entrée, dans ce cas ce sera en Q15. Ensuite, on applique les filtres au signal en faisant une multiplication qui devrait avoir un résultat en format Q2.28. Ensuite, avant de faire la transformée inverse, il faut changer le format Q2.28 à Q15. Pour ce faire, on ignore les 13 LSB et le MSB. La sortie devrait être en format Q15

## Résultats attendus, conditions du test :

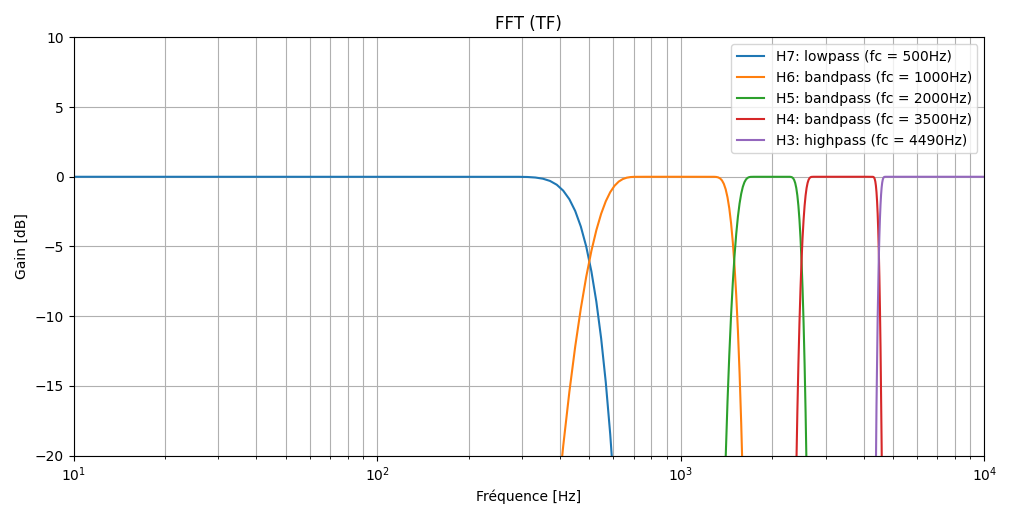


Figure 6 : Réponse fréquentielle des filtres H3 à H7 par python

La réponse fréquentielle de la Figure 6 représente ce qu’il faut atteindre théoriquement. Il est possible de voir chacun des filtres séparés et comment ils se chevauchent.

## Test unitaire à exécuter :

**Python :**

Sur Python les réponses en fréquence ont été analyser pour une onde sinusoïdale de 2000 Hz. Les réponses doivent aussi prendre en considération les formats des nombre Q15 et Q2.13. Il est important puisque pour avoir des réponses similaires que ceux sur le DMCI, il est nécessaire de modifier les formats.

## Résultats obtenus :

**Python :**

Toutes les réponses en fréquence de chaque filtre H3 à H7 ainsi que l’addition de tous les filtre pour une fréquence de 2000 Hz peuvent être retrouver en annexe à la Figure 18.

# B3 & B4 – FFT inverse, extraction de la trame filtrée

## Description :

Pour revenir dans le domaine temporel, une FFT inverse est réalisée pour le signal passé dans chacun des filtres. Le code permet de choisir quels filtres sont actifs avec les interrupteurs SW7 à SW3 et d’additionner leurs sorties en fonction de ceux qui sont choisis.

## Résultats attendus, conditions du test :

Il est possible de tester chacun des filtres en les isolant et en injectant une onde sinusoïdale d’entrée qui varie autour de sa ou de ses fréquences de coupure. Par exemple, en testant le passe-bas, si l’onde est de 300 Hz, son amplitude devrait rester inchangée. À 500 Hz, l’amplitude devrait être réduite d’environ -3 dB et vers 700 Hz l’onde devrait être nulle ou presque.

## Test unitaire à exécuter :

**Passe-bas (H7) :**

Avec le Analog Disocvery 2, mesurer le signal de sortie et activer le filtre passe-bas H7 seulement. Faire varier le signal d’entrée entre 300 Hz et 700 Hz.

**Passe-bande 500 – 1500 Hz (H6) :**

Avec le Analog Disocvery 2, mesurer le signal de sortie et activer le filtre passe-bande H6 seulement. Faire varier le signal d’entrée entre 300 Hz et 1700 Hz.

**Passe-bande 1500 – 2500 Hz (H5) :**

Avec le Analog Disocvery 2, mesurer le signal de sortie et activer le filtre passe-bande H5 seulement. Faire varier le signal d’entrée entre 1300 Hz et 2700 Hz.

**Passe-bande 2500 – 4500 Hz (H4) :**

Avec le Analog Disocvery 2, mesurer le signal de sortie et activer le filtre passe-bande H4 seulement. Faire varier le signal d’entrée entre 2200 Hz et 4700 Hz.

**Passe-haut (H3) :**

Avec le Analog Disocvery 2, mesurer le signal de sortie et activer le filtre passe-haut H3 seulement. Faire varier le signal d’entrée entre 4300 Hz et 4700 Hz.

## Résultats obtenus :

**Passe-bas (H7) :**

**Une image contenant capture d’écran, ligne, texte, Tracé

Description générée automatiquement**

Figure 7 : Sinus 300 Hz dans filtre H7

**Une image contenant texte, capture d’écran, ligne, Tracé

Description générée automatiquement**

Figure 8 : Sinus 500 Hz dans filtre H7

**Une image contenant ligne, texte, Tracé, capture d’écran

Description générée automatiquement**

Figure 9 : Sinus 700 Hz dans filtre H7

Une image contenant ligne, diagramme, Tracé, Parallèle

Description générée automatiquement

Figure 10 : Filtre passe-bas H7

À 300 Hz le voltage crête-à-crête est d’environ 3.3 V ce qui est comme attendu puisque c’est l’onde d’entrée sans réduction de gain. À 500 Hz la tension est maintenant à environ 1.7 V ce qui veut dire qu’à 500 Hz il y a un gain de -5,76 dB. C’est attendu puisque tel que vu dans la Figure 10, à 500 Hz le gain du filtre est de -6 dB. À 700 Hz, le voltage est très petit ce qui est normal puisque c’est dans la bande de coupure du filtre.

**Passe-bande 500 – 1500 Hz (H6) :**

En répétant les étapes tel qu’au filtre passe-bas, on arrive à un voltage presque nul à 300 Hz. À 500 Hz, un voltage d’environ 1.7 V, soit un gain de -6 dB. Le voltage à 1000 Hz est d’environ 3.3 V ce qui équivaut à un gain de 0 dB. À 1500 Hz le voltage est encore d’environ 1.7 V, donc il y a un gain de -6 dB. Finalement, à 1700 Hz le voltage est presque nul.

**Passe-bande 1500 – 2500 Hz (H5) :**

En répétant les étapes tel qu’au filtre passe-bas, on arrive à un voltage presque nul à 1300 Hz. À 1500 Hz, un voltage d’environ 1.7 V, soit un gain de -6 dB. Le voltage à 2000 Hz est d’environ 3.3 V ce qui équivaut à un gain de 0 dB. À 2500 Hz le voltage est encore d’environ 1.7 V, donc il y a un gain de -6 dB. Finalement, à 2700 Hz le voltage est presque nul.

**Passe-bande 2500 – 4500 Hz (H4) :**

En répétant les étapes tel qu’au filtre passe-bas, on arrive à un voltage presque nul à 2300 Hz. À 2500 Hz, un voltage d’environ 1.7 V, soit un gain de -6 dB. Le voltage à 3000 Hz est d’environ 3.3 V ce qui équivaut à un gain de 0 dB. À 4500 Hz le voltage est encore d’environ 1.7 V, donc il y a un gain de -6 dB. Finalement, à 4700 Hz le voltage est presque nul.

**Passe-haut (H3) :**

Pour le filtre passe-haut, on retrouve un voltage presque nul à 4300 Hz et un voltage d’environ 1.7 V à 4490 Hz, donc -6 dB. À 4700 Hz le voltage est environ à 3.3 V.

# C1 – Filtres IIR

## Description :

Un filtre IIR a été utiliser pour un filtrage, plus précisément un IIR de type elliptique d’ordre 4. Ce filtre est un filtre avec une bande passante de 950 Hz à 1050 Hz avec un ronflement maximale de 1 dB dans la bande passante et une atténuation maximale de 70 dB dans la bande atténuée. La fonction signal.ellip a été utiliser sur python pour déterminer les coefficients du filtre.

Par la suite pour appliquer les filtres sur une onde sinusoïdale, des calculs doivent être fait. Les formules sont présentées plus bas soit :

Il est important de noter que la convention pour le signe sur les variables ne sont pas universelle. Il est donc possible que la formule soit inversée, il n’a pas de façon mathématique de trouver cette erreur de signe sauf par l’essai erreur sur le signe. Dans notre cas, le code python nous donne les inversé, il faudra donc changer les + par des -.

## Résultats attendus, conditions du test :

Il est possible de tester ce filtre en l’isolant et en injectant une onde sinusoïdale d’entrée variant autour de sa ou de ses fréquences de coupure. Par exemple, à 1000 Hz, l’onde devrait être nulle ou presque puisque cette fréquence est au centre de la bande passante.

## Test unitaire à exécuter :

**Python :**

Pour tester les valeurs de coefficient des sinus de 900 Hz, 1000 Hz et 1100 Hz ont été fait sur python pour tester si cela fonctionne bien. Le but est de voir une grande diminution dans l’onde sinusoïdale de 1000 Hz.

**MPLAB :**

Une fois la validation des coefficients faites, il est possible d’intégrer le tous dans MPLAB. Cette intégration se fait en étape avec des test identique à ceux de python avec le test à 900 Hz, 1000 Hz et 1100 Hz. L’onde devra être atténuer sur le sinus de 1000 Hz et laisser passer ceux de 900 Hz et 1100 Hz.

**Transition :**

Il faut assurer que le code python qui crée les coefficients est belle et bien capable de transférer les valeurs dans le fichier filterFIRcoeffs.h.

## Résultat obtenu :

**Python :**

De plus, à la Figure 20, il est possible de voir que lorsque les coefficients sont mis sur une base Q2.13 la fréquence de coupure est autour du 1000 Hz. Cependant, lorsque ces mêmes coefficients sont placer sur la base Q2.5, les fréquences que coupures des filtres varient. Il ne fonctionnera donc pas dans les applications réelles puisque les fréquences des 4 filtres ne coïncident pas.

Le résultat des tests de python est présenté à la Figure 19. Dans la figure, il est possible de voir trois ondes soit l’entrée avant le filtre et les deux autres étant la sortie après le filtre IIR à différents Q. En analysant le Q2.13, il est possible de voir que lorsqu’une onde de 900 Hz est mise en entrée il y a très peu de modification (modification au début puisque les variables se remplissent) et décalage d’un échantillon. Il est possible de voir le même résultat sur le sinus de 1100 Hz en Q2.13. Cependant, sur le sinus de 1000 Hz, il est clairement possible de voir qu’en Q2.13 l’atténuation est présente. Elle contient toujours les petites variations du début ainsi que les décalages, mais elle est très atténuée à comparer de son entrée. Pour ce qui est de l’onde Q2.5, l’onde est beaucoup moins belle et elle n’est pas atténuer aux bons endroits. Comme il a été expliquer dans le paragraphe plus haut les fréquences de coupure ne coïncident pas tous sur 1000 Hz, donc lorsqu’on rentre des valeurs différentes on peut voir que les atténuations ne sont pas aux bons endroits.

**MPLAB :**

Les mêmes tests ont été fait sur MPLAB, soit une validation du fonctionnement à 900 Hz, 1000 Hz et 1100 Hz. Après ces tests il a été possible de déterminer que le fonctionnement du filtre est adéquat.

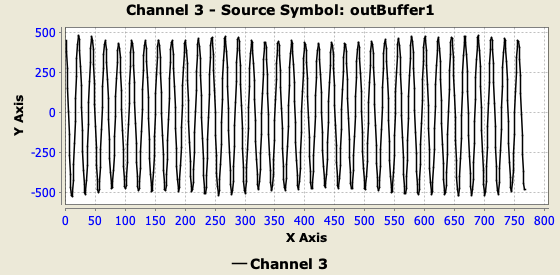


Figure 11 : Sortie du filtre IIR à 900 Hz

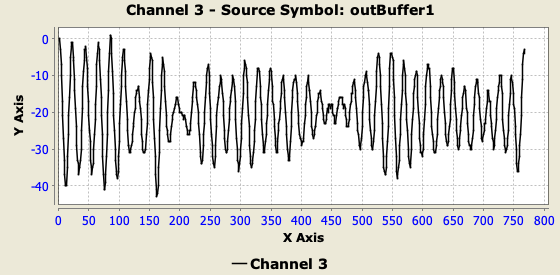


Figure 12 : Sortie du filtre IIR à 1000 Hz

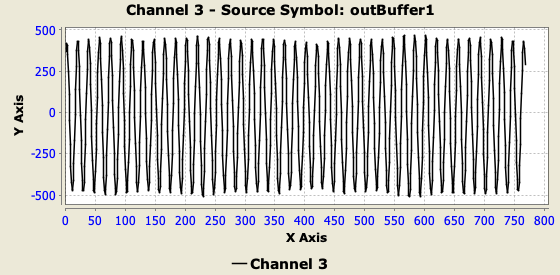


Figure 13 : Sortie du filtre IIR à 1100 Hz

**Transition :**

Dans la Figure 14, il est possible de voir les réponses de python par rapport au filtre IIR au paramètre demander. Dans la Figure 15, il est possible de voir les mêmes chiffres dans la fichier filterFIRcoeffs.h, cela veut donc dire que le transfert a bien été fait.

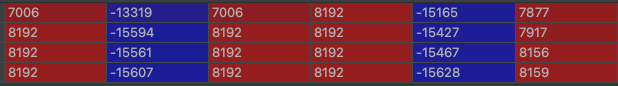
****

Figure 14 : Résultat Python filtre IIR Q2.13

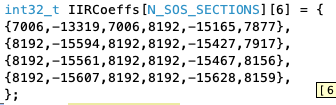
****

Figure 15 : Fichier filterFIRcoeffs.h dans MPLAB filtre IIR

# ANNEXE A : GRAPHIQUE PYTHON FENÊTRAGE

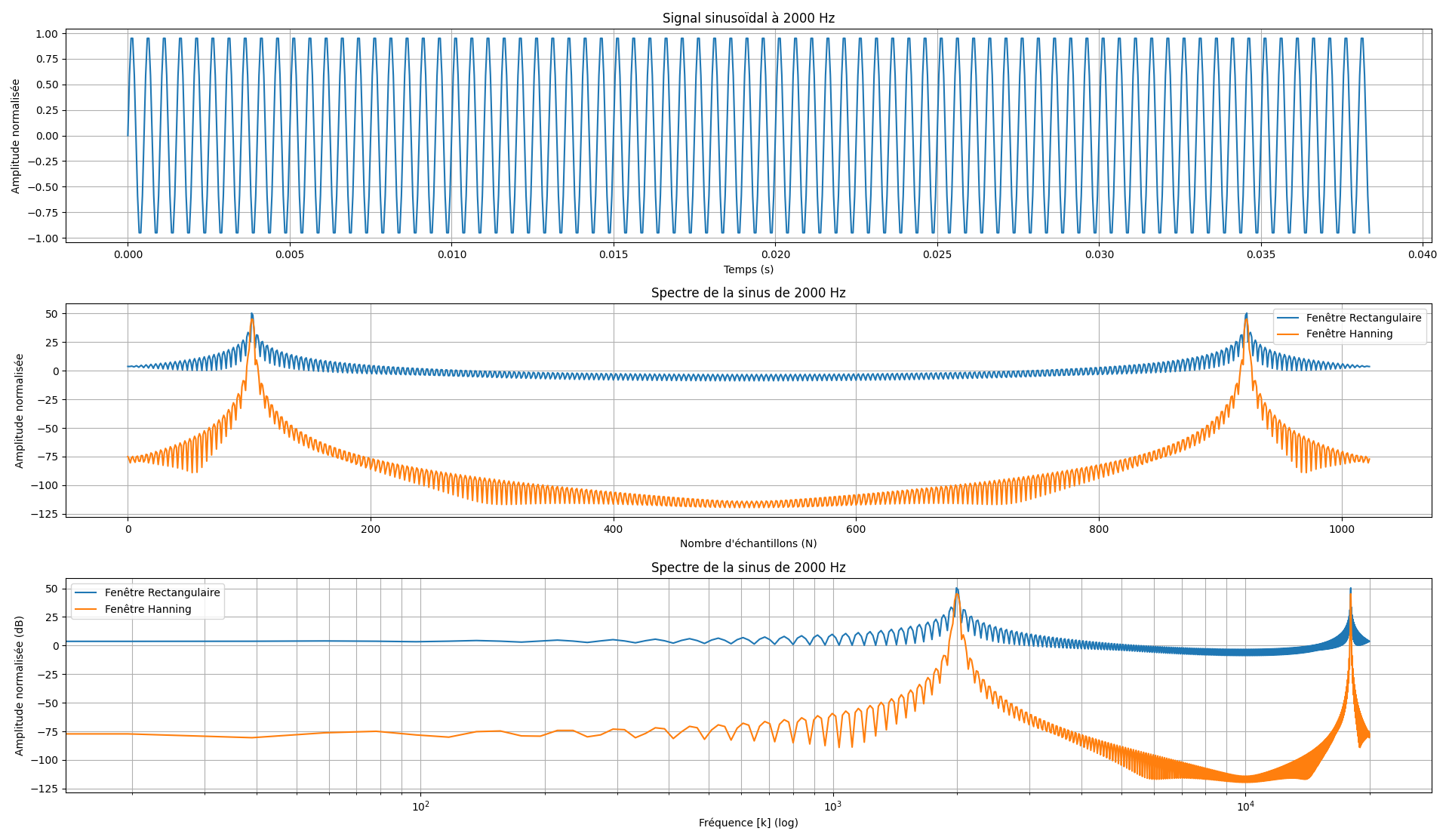


Figure 16 : Graphique Python du spectre de fréquence sur une sinus de 2000 Hz

# ANNEXE B : GRAPHIQUE MPLAB FENÊTRAGE

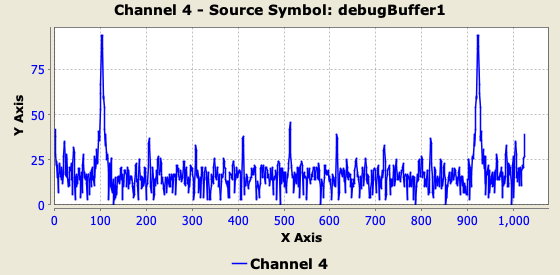
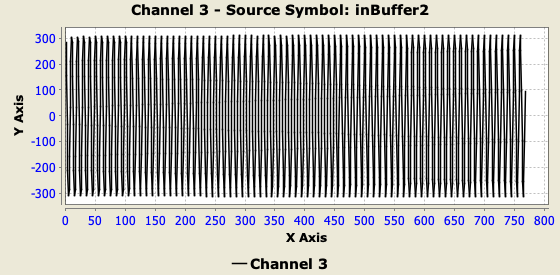
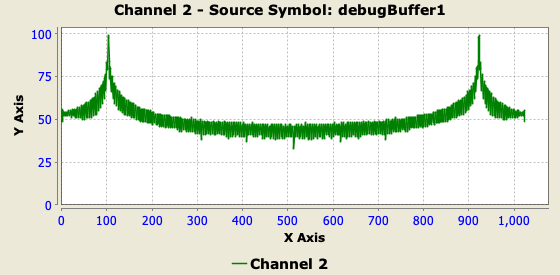
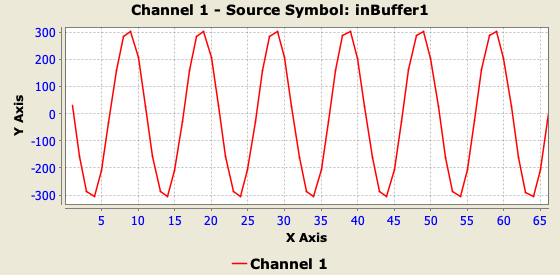


Figure 17 : Graphique DMCI du spectre de fréquence sur une sinus de 2000 Hz

# ANNEXE C : RÉPONSE EN FRÉQUENCES DES FILTRES FIR

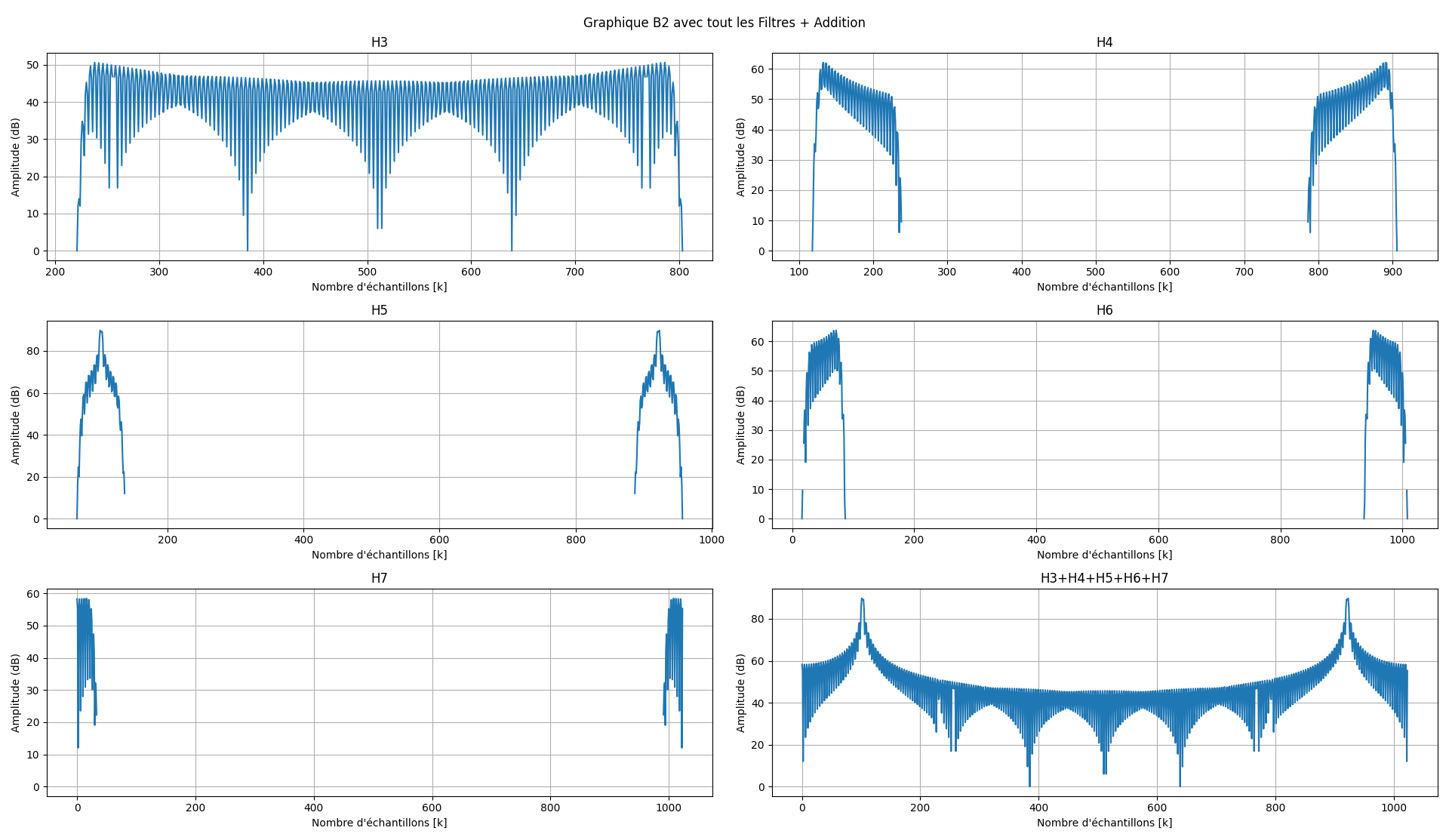


Figure 18 : Graphiques Python des réponses en fréquences des filtres FIR

# ANNEXE D : GRAPHIQUES PYTHON DU FILTRE IIR

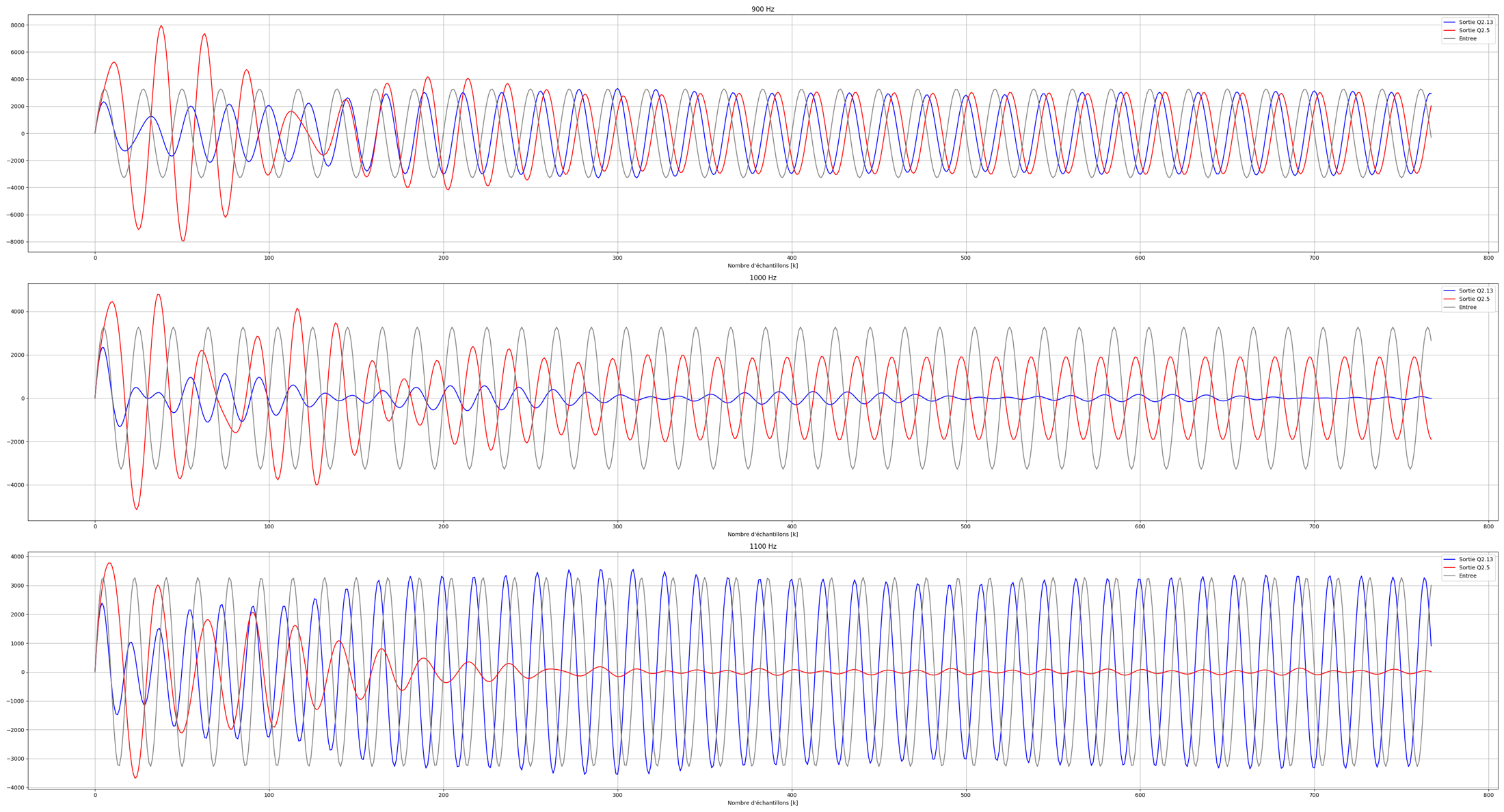


Figure 19 : Graphiques des test python du filtre IIR

# ANNEXE E : FONCTION DE TRANSFERT FILTRE IIR

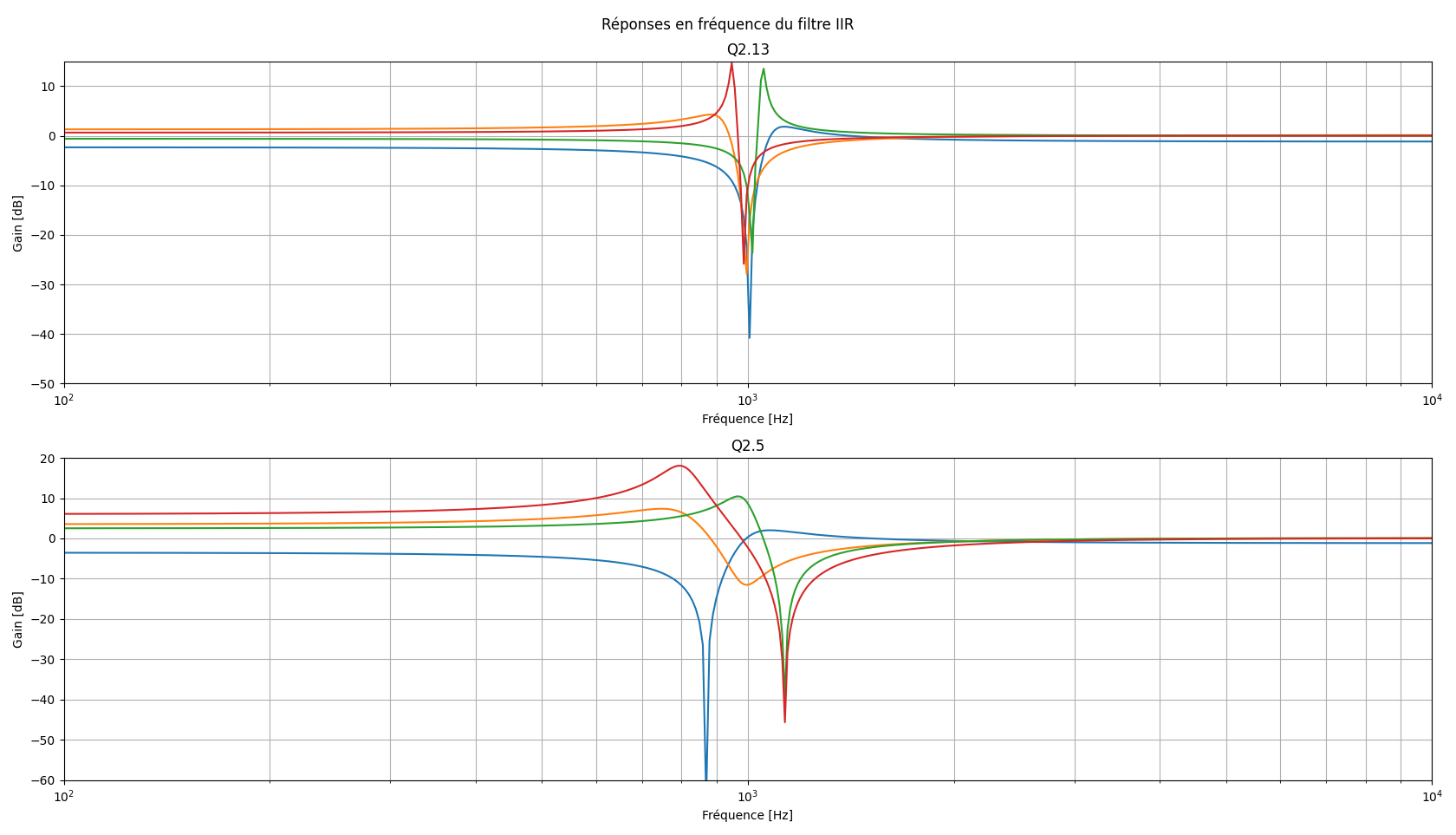


Figure 20 : Fonction de transfert du filtre IIR