Etude de cas: Filtrage Numérique

Dans ce TP, nous allons continuer à nous intéresser à la conception de la chaîne d'acquisition pour le robot dont on a étudié les spécifications de son convertisseur analogique numérique (CAN) durant le dernier TP. Pour rappel, l'étude nous a permis de fixer la fréquence d'échantillonnage du CAN à 250 KHz, sa référence Vref à 4 V et le nombre de bits à 14.

Dans ce TP, nous allons nous concentrer sur le filtrage numérique. Nous allons étudier le filtrage nécessaire pour le signal issu d'un des trois capteurs considérés, le capteur de champs magnétique. Pour rappel, la fréquence de sortie maximale de ce capteur est de 20 KHz avec une dynamique d'entrée de $\pm$ 10 mV à $\pm$ 3 V sur laquelle il faut garantir un SNDR de 40 dB. Ce signal sera perturbé par des interféreurs que nous avons modélisés par deux sinusoïdes d'amplitude 0.5 V à 32 KHz et 80 KHz. [1]

L'objectif du TP est d'étudier et dimensionner le filtrage numérique nécessaire pour passer du signal cadencé à 250 KHz à un signal décimé cadencé à 41.6667 KHz avec un SNDR supérieur à 40 dB. (Vous avez û noter que la fréquence d'échantillonnage après décimation n'est pas exactement 2×20 KHz, nous avons en fait pris 41.6667 KHz pour avoir un rapport entier de 6 avec 250 KHz.) Nous souhaitons également que l'ondulation à l'intérieur de la bande passante [0 20 KHz] soit inférieure à $\pm$ 0.1 dB.

Le SNDR dans la bande utile à la sortie du CAN est de 42 dB pour un sinusoïde à 5 Khz d'amplitude 10 mV.

Question 1 *A quelles fréquences se replient les deux interféreurs après décimation ? Charger et exécuter le fichier* `Filter_decimation.sce` *et comparer les résultats de simulations à vos calculs.*

Pour concevoir le filtre qui nous permettra de réaliser cette décimation tout en minimisant la complexité, nous allons procéder en 3 étapes :

1. Calculer l'ordre et les coefficients du filtre. Nous allons concevoir un filtre FIR symétrique en utilisant une approche numérique détaillé dans la fonction `eqfir`
2. Coder les coefficients en virgule fixe et déterminer le nombre de bits adéquat
3. Coder les coefficients en canonical signed digit (CSD)

Les 2 premières étapes vont chacune causer une dégradation au SNDR. Nous allons ainsi attribuer une dégradation de 0.5 dB pour la première étape (42 à 41.5 dB), une dégradation de 1.5 dB pour la 2ème (41.5 à 40 dB).

1. Sachez qu'en pratique les interféreurs peuvent être aussi bien des raies parasites que des signaux plus riches en fréquence. La forme et l'amplitude de ces interféreurs dépendent d'une multitude de paramètres tels que la nature du capteur, le type d'alimentation utilisée, l'environnement,...

1 Etape 1 : ordre du filtre

Question 2 *Charger et exécuter le fichier* `Filter_order.sce`*. Observer les spectres avant filtrage, après filtrage et après décimation ainsi que la réponse fréquentielle du filtre.*

Question 3 *Déterminer en simulation l'ordre du filtre minimal L_{min} qui permet d'obtenir un SNDR supérieur à la valeur visée pour la première étape et une ondulation inférieure à ± 0.1 d.B*

Question 4 *Relever l'ordre du filtre et tracer sa réponse impulsionelle. Sauvegarder les coefficients du filtre dans un à l'aide de la commande* `save('hn.txt',hn)`

Question 5 *Afin d'éviter de les recalculer dans la suite, sauvegarder les coefficients du filtre dans un fichier à l'aide de la commande* `save('hn.txt',hn)`

2 Etape 2 : codage en virgule fixe (quantification)

Nous allons à présent coder les coefficients du filtre en virgule fixe ou en d'autres termes nous allons quantifier les coefficients du filtre sur un nombre fini de bits.

Question 6 *Charger et exécuter le fichier* `Filter_NumOfBits.sce`*. Comparer les réponses fréquentielles du filtre avant et aprés quantification.*

Question 7 *Déterminer en simulation le nombre de bits minimal $nbitsfilter_{min}$ qui permet d'obtenir un SNDR supérieur à la valeur visée pour la deuxième étape.*

Question 8 *Afin d'éviter de les recalculer dans la suite, sauvegarder les coefficients quantifés du filtre dans un fichier à l'aide de la commande* `save('hn_quant.txt',hn_quant)`

3 Etape 3 : codage en CSD

Afin de minimiser la complexité du filtre, nous allons l'implémenter à l'aide d'une architecture sans multiplieur. Nous allons en fait utiliser une approche avec des décalages et des sommes. Par exemple, pour implémenter une multiplication par 7, on pourrait faire un premier décalage de l'entrée du multiplieur de 2 bits vers la gauche pour implémenter une multiplication par 4, un deuxième décalage d'un bit pour implémenter une multiplication par 2 et ensuite sommer les 2 versions décalées avec l'entrée originale. Avec l'approche CSD, la multiplication par 7 se fera par un décalage de 3 bits vers la gauche pour implémenter une multiplication par 8 suivi d'une soustraction de l'entrée originale de cette version décalée. Ceci ainsi permet de réduire la complexité de la multiplication.

Question 9 *Pour mieux comprendre le fonctionnement du codage en CSD, charger le fichier* `CSD_basic.sce`*. Changer la valeur de in par exemple 7, 54, 127 ... Comparer les sorties binaires aux sorties CSD.*

Question 10 *Pour Calculer la complexité du filtre conçu, nous allons calculer le nombre de bits non nuls dans les coefficients. Charger le fichier* `Filter_complexity.sce`*, modifier la valeur de $nbitsfilter$ à la valeur que vous avez retenue dans la section précédente. Relever le nombre de sommateur/soustracteur en binaire et CSD, comparer avec les résultats de vos collègues.*