

# TP 3- FILTRES NUMERIQUES IIR

## 1. Introduction :

Un système de traitement numérique de signal fournit à chaque instant  $nT$ , où  $T$  est la période d'échantillonnage, une sortie  $y(nT)$  pour une entrée  $x(nT)$ . Ce traitement temps réel est possible avec des processeurs spécialisés. Pour simplifier l'écriture on peut omettre la variable  $T$  (on écrit  $n$  au lieu de  $nT$ )

L'objectif de ce TP est de dimensionner et implémenter un filtre **IIR** (Infinite Impulse Response) sur un microcontrôleur STM32F4.



Tout échantillon du signal sortie est une somme pondérée des échantillons d'entrée et de ses états antérieurs.

$$y(n) = \sum_{k=0}^M b(k) \cdot x(n-k) - \sum_{k=0}^N a(k) \cdot y(n-k) ; \text{ où } b(k) \text{ et } a(k) \text{ les coefficients du filtre.}$$

## 2. Synthèse du filtre IIR :

On souhaite programmer un filtre passe bas numérique à réponse impulsionnelle infinie «IIR» ayant les spécifications suivantes :

Fréquence de coupure à  $A_c = -3dB$  égale à  $f_c = 1000Hz$

Fréquence d'atténuation à  $A_a = -20dB$  égale à  $f_a = 3000Hz$

Fréquence d'échantillonnage  $f_s = 10KHz$

1. Calculer les pulsations  $\omega_c$  et  $\omega_a$  du filtre numérique.
2. Faites la distorsion de fréquence pour déterminer les pulsations  $\omega_{ac}$  et  $\omega_{aa}$  du filtre analogique équivalent.
3. Calculer alors l'ordre du filtre.
4. Pour  $n = 2$  ; déterminer la fonction de transfert du filtre passe bas de Butterworth normalisé  $H_p(s)$ .
5. Dénormaliser la fonction de transfert déjà obtenue.
6. Appliquer la transformation bilinéaire pour calculer la fonction de transfert  $H(z) = \frac{Y(z)}{X(z)}$ .
7. Déduire alors l'équation différentielle  $y(n)$ .

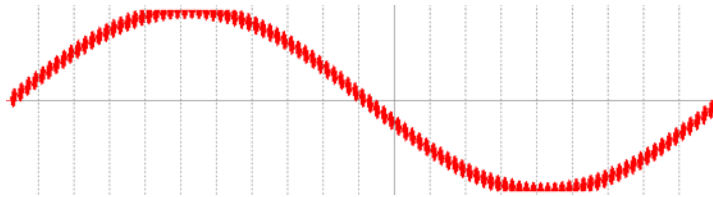
### 3. Programmation du filtre IIR :

La synthèse du filtre par la méthode bilinéaire du filtre de Butterworth  $H(z) = \frac{Y(z)}{X(z)}$  a donné la fonction de

transfert suivante :  $H(z) = \frac{0,0674 + 0,1348z^{-1} + 0,0674z^{-2}}{1 - 1,143 \cdot z^{-1} + 0,412z^{-2}}$

1. Déterminer les coefficients  $b(k)$  et  $a(k)$  du filtre récursif.
2. Ouvrir le projet *D:\FIR1\MDK-ARM\IIR.uvprojx*, dans le fichier *IIR.h*, initialiser les tableaux **B** et **A** par les coefficients  $b(k)$  et  $a(k)$ . Les tableaux **XX** et **YY** sont déjà initialisés par des zéros.

Dans notre cas, le signal d'entrée est une sinusoïde bruitée dont les échantillons sont stockés dans un tableau **noisy\_sine** de 1000 échantillons.



On vous demande de compléter le code de la fonction du filtre IIR « *float32\_t IIR (float32\_t x)* » ayant comme paramètre d'entrée la valeur d'un échantillon du signal d'entrée et renvoie la valeur du signal de sortie correspondant.

```
float32_t IIR(float32_t Xin)
{
    int16_t i;

    // Décaler les éléments du tableau XX.
    // Décaler les éléments du tableau YY.

    XX[0] = Xin ;
    y = 0;

    // calculer alors la valeur du signal de sortie y.

    YY[0] = y ;
    return (y);
}
```

Dans le programme principal, cette fonction est appelée toutes les dixième de millisecondes (vous pouvez profiter de la routine de l'interruption du timer **SysTick**).

```
while (1)
{
    if(TeFlag == 1)
    {
        TeFlag = 0;
        Xin = noisy_sine[i] ;

        Yout = IIR(Xin);
        j++;
        if(j > 1000)
            while(1);
    }
}
```

Compiler et déboguer votre programme, utiliser l'analyseur logique pour afficher les valeurs des variables **Xin** et **Yout**.