

TP 2- FILTRES NUMERIQUES FIR

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 d'implémenter un filtre **FIR** (Finite Impulse Response) sur un microcontrôleur STM32F4.



Tout échantillon du signal sortie est une somme pondérée des échantillons d'entrée.

$$y(n) = \sum_{k=0}^{K-1} b(k) \cdot x(n-k)$$
 ; où $b(k)$ est le coefficient du filtre d'ordre k et $x(n-k)$ échantillon antérieur d'ordre k du signal d'entrée.

Le tableau suivant illustre de façon simple la détermination des échantillons de sortie.

	suivant ←					Maintenant	→ précédent							
k	-5	-4	-3	-2	-1	0	1	2	3	4	10		
$b(k)$	0	0	0	0	0	$b(0)$	$b(1)$	$b(2)$	$b(3)$	$b(4)$	$b(10)$		
$x(-k)$	$x(5)$	$x(4)$	$x(3)$	$x(2)$	$x(1)$	$x(0)$	0	0	0	0	0	0	$y(0)$
$x(1-k)$	$x(6)$	$x(5)$	$x(4)$	$x(3)$	$x(2)$	$x(1)$	$x(0)$	0	0	0		0	0	$y(1)$
$x(2-k)$	$x(7)$	$x(6)$	$x(5)$	$x(4)$	$x(3)$	$x(2)$	$x(1)$	$x(0)$	0	0	0	0	$y(2)$
\vdots						\vdots							0	\vdots
$x(11-k)$	$x(15)$	$x(14)$	$x(13)$	$x(12)$	$x(11)$	$x(10)$	$x(9)$	$x(8)$	$x(7)$	$x(6)$	$x(0)$	0	$y(11)$
$x(12-k)$	$x(16)$	$x(15)$	$x(14)$	$x(13)$	$x(12)$	$x(11)$	$x(10)$	$x(9)$	$x(8)$	$x(7)$	$x(1)$	$x(0)$	$y(12)$

2. Implémentation du filtre FIR :

On souhaite programmer un filtre numérique à réponse impulsionnelle finie « FIR » ayant les caractéristiques suivantes :

$$f_c = 1000\text{Hz} ; f_s = 10\text{KHz} ; N = 10$$

Le filtre FIR a pour équation : $y(k) = b_0x(k) + b_1x(k-1) + b_2x(k-2) + \dots + b_{10}x(k-10)$

Pour la détermination des coefficients du filtre, nous allons utiliser la fonction standard **fir1** de Matlab/GNU Octave.

1) Lancez le logiciel **GNU Octave (GUI)** et tapez dans la fenêtre de commande :

```

>> pkg load signal      % chargement du paquet signal contenant la fonction fir1
>> fc= 1000;            % fréquence de coupure à -3 dB
>> fs = 10000           % fréquence d'échantillonnage
>> wc = fc/(fs/2);      % fréquence de coupure normalisée à fs/2
>> b = fir1(10, wc);    % cette fonction calcule les coefficients du filtre
                        % FIR :
                        % paramètres :
                        %      10 : Ordre du filtre
                        %      wc : fréquence de coupure normalisée
                        % b : tableau des coefficients du filtre : b0, b1, ...

```

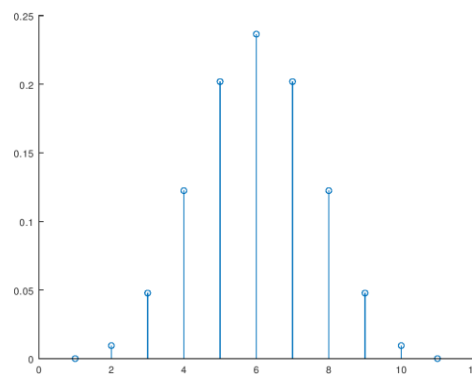
La fonction **fir1** retourne, les 11 coefficients du filtre (b_0, b_1, \dots, b_{10}) dans un tableau **b**.

Avant de passer à la programmation tapez la commande suivante pour voir la réponse impulsionnelle fréquentielle du filtre :

```

>> figure; stem(b) ;

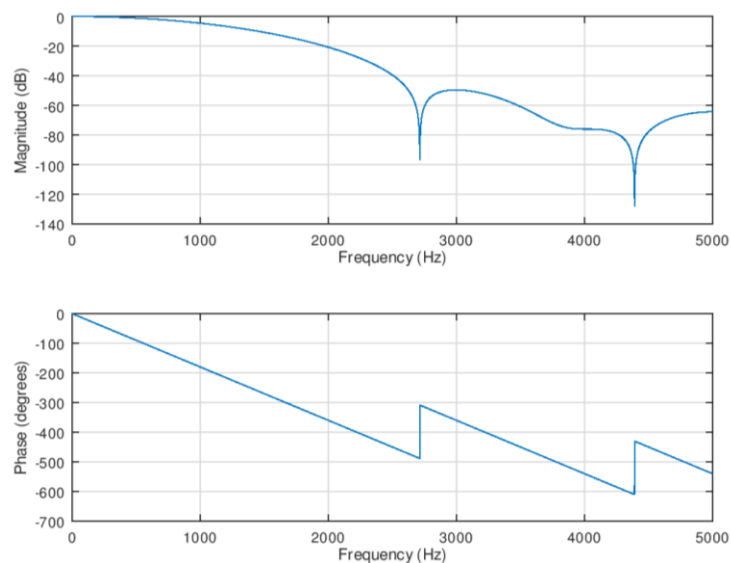
```



```

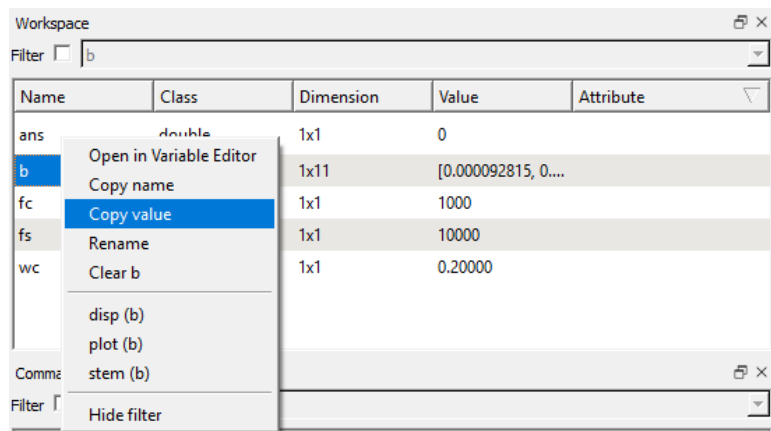
>> freqz(b,1,2^12,fs)  % b : coef. du filtre (numérateur)
                        % 1 : (dénominateur = 1)
                        % 2^12 : nombre de point (par défaut : 512)
                        % fs : fréquence d'échantillonnage

```



3. Programmation du filtre FIR :

Dans la fenêtre **Workspace**, faites un clic droit sur la variable **b** pour copier les coefficients du filtre dans le presse-papier.

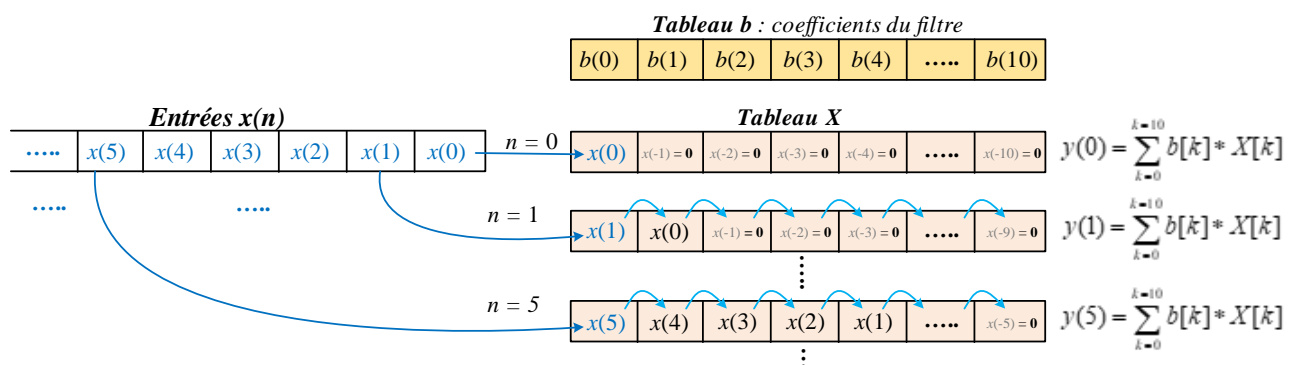


Ouvrir le projet **D:\FIR1\MDK-ARM\FIR1.uvprojx**, et coller les coefficients du filtres dans le tableau **b** du fichier **fir.h**.

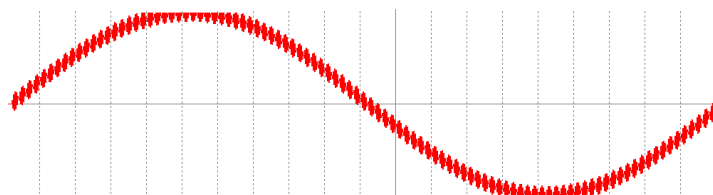
Le tableau **X** initialisée à 0 est utilisé pour sauvegarder les échantillons antérieurs du signal d'entrée $x(n)$.

A chaque fois qu'on veut calculer un échantillon de sortie $y(n)$, il faut :

- Décaler les éléments du tableau **X** vers la droite (voir figure ci-dessous).
- Charger la première case du tableau **X** par le nouveau échantillon d'entrée $x(n)$.
- Calculer la valeur de l'échantillon de sortie $y(n) = \sum_{k=0}^{k=10} b[k] * X[k]$ où $n = 1, 2, 3, \dots$



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 FIR « **float32_t FIR (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.

```

float FIR(float Xin)
{
    int16_t i;

    // Décaler les éléments du tableau X.

    X[0] = Xin ;
    y = 0;

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

    return (y);
}

```

Dans le programme principal, cette fonction est appelée toutes les 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 = Fir(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**.

4. Traitement d'un signal sonore

On veut maintenant appliquer le filtre étudié précédemment à un signal sonore. Le message sonore est enregistré dans un fichier **v28_mes4.wav**.

Revenons au logiciel GNU Octave.

1. Sélectionner dans la barre de menu **Current Directory** le dossier **D:\FIR1\wave**.
2. Ouvrir un nouveau fichier script et enregistrer là sous le nom **fir.m**
3. Tapez dans ce fichier les instructions suivantes :

```

% ouverture du fichier wav
file = 'v28_mes4.wav';
[s_data , fs] = audioread(file);

% Détermination des coefficients du filtre
pkg load signal
fc = 700;
wc = fc/(fs/2);
global b = fir1(19,wc);

% Filtrage du message sonore
xx1 = zeros(20,1);
out = zeros(65536,1);
for n = 1:65535
    varr = s_data(n);
    for i = 1:19

```

```

    xx1(21-i) = xx1(20-i);
end;
xx1(1) = varr;
y = 0;
for i = 1:20
    y = y +xx1(i)*b(i);
end;
out(n) = y;
end;

% enregistrement du message filtré

filew = 'fir_mes.wav';
audiowrite(filew, out, fs);

% Affichage
t = linspace(0,1, length(s_data));
figure ; plot(t,s_data);
Nfft = 1024;
f = linspace(0, fs, Nfft);
Gin = abs(fft(s_data,Nfft));
Gout = abs(fft(out,Nfft));
figure;plot(f(1:Nfft/2), Gin(1:Nfft/2));
figure;plot(f(1:Nfft/2), Gout(1:Nfft/2));

```

4. Enregistrer à nouveau le fichier fir.m
5. Tapez dans la fenêtre de commande :

```
>> fir
```

6. Tapez dans la fenêtre de commande les commandes suivantes :

```

>> t = linspace(0,1, length(s_data));
>> figure ; plot(t,s_data);

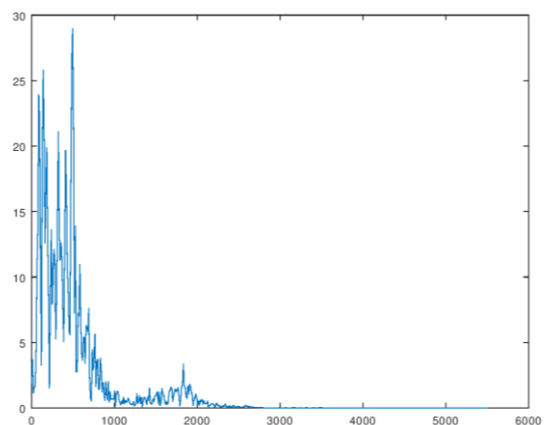
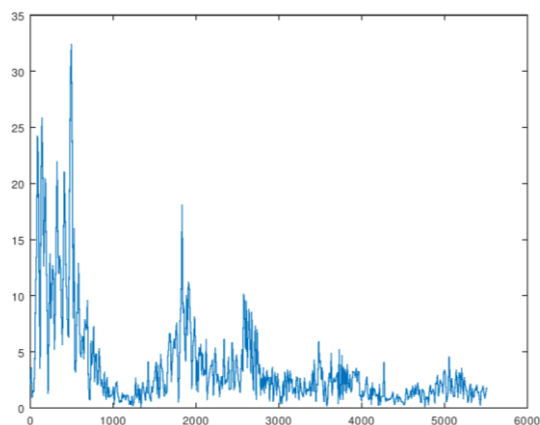
```

7. Affichage du spectre de deux signaux, tapez alors les commandes suivantes :

```

Nfft = 1024
f = linspace(0, fs, Nfft);
Gin = abs(fft(s_data,Nfft));
Gout = abs(fft(out,Nfft));
figure;plot(f(1:Nfft/2), Gin(1:Nfft/2));
figure;plot(f(1:Nfft/2), Gout(1:Nfft/2));

```



8. Ecouter le deux fichiers v28_mes4.wav et fir_mes.wav et conclure.