EXPERIMENT 12

Implementation of FIR Filters on MATLAB and TMS320C6713

Objective

In this lab you will learn how to implement the FIR filter on MATLAB and how to implement an FIR (finite impulse response) filter on C6713 DSP Board.

Introduction

FIR filters are frequently used in the real time DSP systems. They are simple to implement, stable and have property of linear phase. Input and output relationship is given by:

$$\boldsymbol{y[n] = \sum_{m=0}^{M-1} h[m].\,x[n-m]}$$

Where,

$$x = input, y = output, h = filter\ coefficients, M = \ number\ of\ filter\ coefficients.$$

1. Implementation of FIR filter on C6713 DSP Board

1.1 Quantization Consideration:

The key choice in Quantization consideration is between the **floating points** and**fixed point.**

As we are using C6713 in our lab which is a floating point processor and allows using the C language, so it is our ultimate choice. Advantages of the floating point math are:

- Less quantization error
- Don't have to worry about scaling factors
- Less likelihood of overflow/underflow
- Much easier to code

1.2 Code for filter realization:

Direct-Form 1 implies the direct realization of the convolution equation:

$$y[n] = \sum_{m=0}^{M-1} h[m].x[n-m]$$

Allocate buffer of length M for input samples.

1.3 Sample code

```
Interrupt void serialPortRcvISR()
{
union {Uint32 combo; short channel[2];} temp;
inti = 0;
float result = 0.0;
temp.combo = MCBSP_read(DSK6713_AIC23_DATAHANDLE);
// Update array samples (move data - this is the slow way)
for(i = N-1; i>= 1; i-- )
samples[i] = samples[i-1];
samples[0] = (float)temp.channel[0]; // store right channel
// Filtering
for(i = 0 ; i< N ; i++ )
result += fir_coeff[i]*samples[i];
temp.channel[0] = (short)result; // output to right channel
MCBSP_write(DSK6713_AIC23_DATAHANDLE, temp.combo);
}
```

Note that all math here is floating point. Filter coefficients are also assumed to be floating point

Exercise 1:

Create an FIR filter with the following specifications:

- Band pass
- 8th order
- Direct Form I

- Least-squares design
- 44100Hz sampling rate
- Fstop1=3000Hz
- Fpass1=4000Hz
- Fpass2=8000Hz
- Fstop2=12000Hz
- Equal weighting in all bands
- All floating point math (single or double precision)

Show your working and code in the space provided below:

2. Implementation of FIR filter in MATLAB

Finite impulse response (FIR) filters can be characterized by a linear, constant-coefficient difference equation (N is an integer):

$$y(n) = b_0x(n)+ b_1x(n-1)+ b_2x(n-2)+...+ b_Lx(n-N+1)$$

The ideal impulse response for frequency-selective filters is noncausal and infinite-durational. Perhaps the simplest way to design an FIR filter is to truncate the ideal impulse response to obtain a causal, finite impulse response. The window function is the signal which, when multiplied with the infinite impulse response, renders the latter finite i.e.

Ideal impulse response:sin $\left(\frac{\omega cn-\omega cnd}{\pi n-\pi nd}\right)$

Ideal impulse response: $h_{lp}(n) = \sin(\omega_c n-\omega_c n_d)/(\pi n-\pi n_d)$ $(-\infty<n<\infty)$

Windowing function: $w(n) = \begin{cases} 1, & 0 \le n < N \\ 0, & otherwise \end{cases}$

Finite impulse response: $h_{firlp}(n) = w(n)*h_{lp}(n) = \sin(\omega_c n-\omega_c n_d)/(\pi n-\pi n_d)$ $(0<=n<N)$

$H_{lp}(e^{j\omega}) = \begin{cases} 1 & 0 \le |\omega| < \omega c \\ 0 & \omega c < \omega \le \pi \end{cases}$

Figure 12.1 shows the ideal filter response.

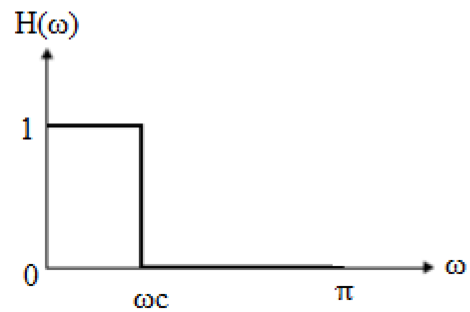

Figure 12.1 - Ideal low-pass filter frequency response

a) Write a script that implements a low-pass FIR filter,where:

$$h(n) = sinc(\omega c * n)$$

$$x(n) = \cos(\omega n)$$

$$y(n) = h(n) * x(n)$$

Plot h(n),H(w),x(n),X(w) and Y(w),where Y(w)=H(w)*X(w).

Fs=32000 samples/sec,Fc=4000Hz,n=-20:20.

b) Prove, analytically, the following theorems given that the Fourier transform of x(n) is $X(e^{jw})$ (i.e. x(n) <-> $X(e^{jw})$).

$$x(n-d) = e^{-j\omega d}X(e^{j\omega})$$

d=5

c) If $h(n)_{lp}$ denotes the impulse response of low pass filter with frequency response of a low pass $H_{lp}(w)$, a high pass filter cab be obtained by translating $H_{lp}(w)$by π radians.

$$H_{hp}(\omega)=H_{lp}(\omega-\pi)$$

Transform the above low pass filter to high pass filter.

Show your working and code in the space provided below: