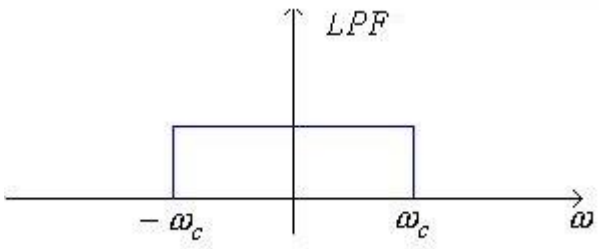
Communications and Signal Processing Lab

Assignment No.-2

EE21MTECH14002

1. Low Pass Filter (or Half Band Filter):

A low-pass filter is a filter that passes signals with a frequency lower than a selected cutoff frequency and attenuates signals with frequencies higher than the cutoff frequency. The filter is sometimes known as **high-cut filter**, or **treble-cut filter** in audio applications. A low-pass filter is the complement of a high-pass filter.



Designing of Low Pass Filter (LPF):

The simplest method for the designing of the finite impulse response filter is known as windowing method.

Step1: Let the desired ideal frequency response of a low pass filter is $H_d(e^{jw})$.

Step2: Take IFFT of $H_d(e^{jw})$ to get $h_d[n]$.

Step3: Since $h_d[n]$ has infinite length, truncate it using a finite length window function w[n] to get h[n].

$$\mathbf{h[n] = h_d[n] \times w[n]}$$

Step4: We can see our practical filter frequency response by taking FFT of h[n] which is $H(e^{jw})$ and you can plot magnitude and phase response.

Low Pass Filter Using C-Programming:

```
void Low_pass_Filter(double hd1_positive_side[],double hd1_negative_side[], double w1[], double h1[],int N1, double wc1, int fc1)
{
      hd1_positive_side[0]=wc1/3.14;

for(int n=1; n<=(N1-1)/2;n++)
    {
      hd1_negative_side[n]=(sin(wc1*-n*0.01744))/(3.14*-n);
      hd1_positive_side[n]=(sin(wc1*n*0.01744))/(3.14*n);
    }

for(int n=0; n<=N1-1;n++)
    {
      w1[n]=0.54-(0.46*cos((2*3.14*n*0.01744)/(N1-1)));
    }

}
```

Above code depict the method to design a low pass filter using windowing method with c-programming.

Since in c-programming indexing is from 0 to n. So, I take two different arrays to store $\mathbf{h_{d1}[n]}$ values i.e., positive and negative side, **w1[n]** is a array for windowing function we used hamming window in this example and **wc1**, **fc1** is the cut-off frequencies in radians/sec and Hz respectively. Then using a for loop, I entered values in positive and negative array based upon IFFT.

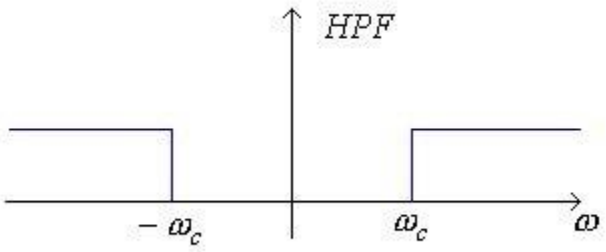
$$h_d[n] = \begin{cases} \dfrac{Sin(\omega_c\, n)}{\pi\, n}, & -(N-1)/2 \le n \le (N-1)/2 \\ \dfrac{\omega_c}{\pi}, & n = 0 \end{cases}$$

Again, by using another for loop to enter values in my hamming window function and then by multiplying $\mathbf{h_d[n]}$ with **w1[n]** we get out desired finite response of low pass filter i.e., **h1[n].**

Since, In c-code trigonometric function take only radians value that's why I multiplies every degree of sine or cosine with **0.0174**(or $\frac{\pi}{180}$**)** to convert it into radians.

2. High Pass Filter (HPF):

A high-pass filter (HPF) is an electronic filter that passes signals with a frequency higher than a certain cutoff frequency and attenuates signals with frequencies lower than the cutoff frequency. It is also known as **low-cut filter** or **bass-cut filter** in the context of audio engineering.



High-pass filters have many uses, such as:

1. Blocking DC from circuitry sensitive to non-zero average voltages or radio frequency devices.
2. Used to remove unwanted sounds..
3. They can also be used in conjunction with a low-pass filter to produce a bandpass filter.

Designing of High Pass Filter(HPF):

The simplest method for the designing of the high pass filter is known as windowing method.

Step1: Let the desired ideal frequency response of a high pass filter is $H_d(e^{jw})$.

Step2: Take IFFT of $H_d(e^{jw})$ to get $h_d[n]$.

Step3: Since $h_d[n]$ has infinite length, truncate it using a finite length window function w[n] to get h[n].

$$\mathbf{h[n] = h_d[n] \times w[n]}$$

Step4: We can see our practical filter frequency response by taking FFT of h[n] which is $H(e^{jw})$ and you can plot magnitude and phase response.

High Pass Filter Using C-Programming:

```
void High_pass_Filter(double hd2_positive_side[],double hd2_negative_side[], double w2[], double h2[],int N2, int fs2, double wc2, int fc2)
{
   hd2_positive_side[0]=1-(wc2/3.14);

for(int n=1; n<=(N2-1)/2;n++)
    {
      hd2_negative_side[n]=(sin(3.14*-n*0.01744)/(3.14*-n))-((sin(wc2*-n*0.01744))/(3.14*-n));
      hd2_positive_side[n]=(sin(3.14*n*0.01744)/(3.14*n))-((sin(wc2*n*0.01744))/(3.14*n));
    }

for(int n=0; n<=N2-1;n++)
    {
      w2[n]=0.54-(0.46*cos((2*3.14*n*0.01744)/(N2-1)));
    }
}
```

Above code depict the method to design a high pass filter using windowing method with c-programming.

Since in c-programming indexing is from 0 to n. So, I take two different arrays to store **hd2[n]** values i.e., positive and negative side, **w2[n]** is a array for windowing function we used hamming window in this example and **wc2, fc2** is the cut-off frequencies in radians/sec and Hz respectively and **fs2** is a sampling frequency. Then using a for loop, I entered values in positive and negative array based upon IFFT.

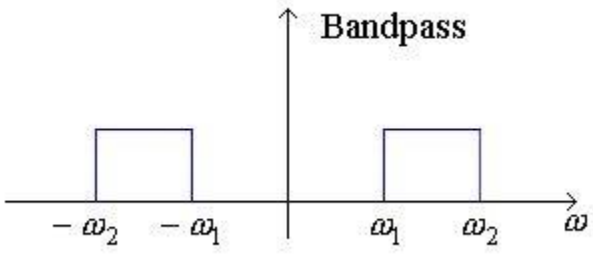
$$h_d[n] = \begin{cases} \dfrac{Sin(\pi n)}{\pi n} - \dfrac{Sin(\omega_c\, n)}{\pi n}, & -(N-1)/2 \le n \le (N-1)/2 \\ 1 - \dfrac{\omega_c}{\pi}, & n = 0 \end{cases}$$

Again, by using another for loop to enter values in my hamming window function and then by multiplying **hd2[n]** with **w2[n]** we get out desired finite response of high pass filter i.e**., h2[n].**

Since, In c-code trigonometric function take only radians value that's why I multiplies every degree of sine or cosine with **0.0174(or π/180)** to convert it into radians.

3. Band Pass Filter (BPF):

A bandpass filter (BPF) is a device that passes frequencies within a certain range and rejects (attenuates) frequencies outside that range. Look at the picture below w1 and w2 are the two cut off frequencies any signal outside this frequency range will be rejected by this band-pass filter.



Band-pass filters have many uses, such as:

1. The main function of this filter in a transmitter is to limit the bandwidth of the o/p signal to the band allotted for the transmission. This avoids the transmitter from interfering with further stations.

2. In a receiver, a bandpass filter allows signals within a selected range of frequencies to be heard or decoded, while preventing signals at unwanted frequencies from getting through. A bandpass filter also optimizes the signal-to-noise ratio and sensitivity of a receiver.

3. Bandpass filters are used in all types of instruments as well as in Sonar, Seismology and even medical applications like EEGs and Electrocardiograms.

4. These filters are also extensively used in optics like lasers, LIDARS, etc.

Designing of Band Pass Filter(HPF):

The simplest method for the designing of the band pass filter is known as windowing method.

Step1: Let the desired ideal frequency response of a band pass filter is $H_d(e^{jw})$.

Step2: Take IFFT of $H_d(e^{jw})$ to get $h_d[n]$.

Step3: Since $h_d[n]$ has infinite length, truncate it using a finite length window function w[n] to get h[n].

$$\mathbf{h[n] = h_d[n] \times w[n]}$$

Step4: We can see our practical filter frequency response by taking FFT of h[n] which is $H(e^{jw})$ and you can plot magnitude and phase response.

Band Pass Filter Using C-Programming:

```
void Band_pass_Filter(double hd3_positive_side[],double hd3_negative_side[], double w3[],
                      double h3[],int N3, double fs3, double wc3_1, double wc3_2, double fc3_1, double fc3_2 )
{

 hd3_positive_side[0]=(wc3_2-wc3_1)/3.14;

for(int n=1; n<=(N3-1)/2;n++)
    {
      hd3_negative_side[n]=(sin(wc3_2*-n*0.01744)-(sin(wc3_1*-n*0.01744)))/(3.14*-n);
      hd3_positive_side[n]=(sin(wc3_2*n*0.01744)-(sin(wc3_1*n*0.01744)))/(3.14*n);
    }

for(int n=0; n<=N3-1;n++)
    {
      w3[n]=0.54-(0.46*cos((2*3.14*n*0.01744)/(N3-1)));
    }



}
```

Above code depict the method to design a band pass filter using windowing method with c-programming.

Since in c-programming indexing is from 0 to n. So, I take two different arrays to store **hd3[n]** values i.e., positive and negative side, **w3[n]** is a array for windowing function we used hamming window in this example and **wc3_1, fc3_1, wc3_2, and fc3_2** is the lower and upper cut-off frequencies in radians/sec and Hz respectively and **fs3** is a sampling frequency. Then using a for loop, I entered values in positive and negative array based upon IFFT.

$$h_d[n] = \begin{cases} \frac{Sin(\omega_{c2}n)}{\pi n} - \frac{Sin(\omega_{c1}n)}{\pi n}, & -(N-1)/2 \le n \le (N-1)/2 \\ \frac{\omega_{c2} - \omega_{c1}}{\pi}, & n = 0 \end{cases}$$

Again, by using another for loop to enter values in my hamming window function and then by multiplying **hd3[n]** with **w3[n]** we get out desired finite response of band pass filter i.e**., h3[n].**

Since, In c-code trigonometric function take only radians value that's why I multiplies every degree of sine or cosine with **0.0174(or π/180)** to convert it into radians.

-----*----------*----------*END----------*----------*----