4 Digital Filtering

In this chapter, we'll study digital filtering methods. Specifically, we'll look into the following:

- Filter specifications
- Filtering in frequency domain
- Filtering in time domain
- Simple filter design – Sum and difference (SD) filters
- Finite Impulse Response (FIR) filters
- Infinite Impulse Response (IIR) filters (using MATLAB functions)

4.1 Filter Specifications

Filtering is the process of keeping components of the signal with certain desired frequencies and removing components of the signal with certain undesired frequencies. Very often, we keep the gain of the required frequency components to 1 or close to 1 and the gain of the undesired frequency components will be 0 or close to 0. In general, there are 4 types of filter: low-pass filter (LPF), high-pass-filter (HPF), band-pass filter (BPF) and band-stop filter (BSF). Each filter will have specific characteristics:

- Passband – the range of frequency components that are allowed to pass
- Stopband – the range of frequency components that are suppressed
- Passband ripple – ripples in the passband, the maximum amount by which attenuation in the passband may deviate from gain (which is normally 1)
- Stopband ripple – ripples in the stopband, the maximum amount by which attenuation in the stopband may deviate from gain (which is normally 0)
- Stopband attenuation – the minimum amount by which frequency components in the stopband are attenuated
- Transition band – the band between the passband and the stopband.

Magnitude frequency responses of ideal filters are shown in Figure 1 where *fc* is the cut-off frequency with *Fs* as the sampling frequency.

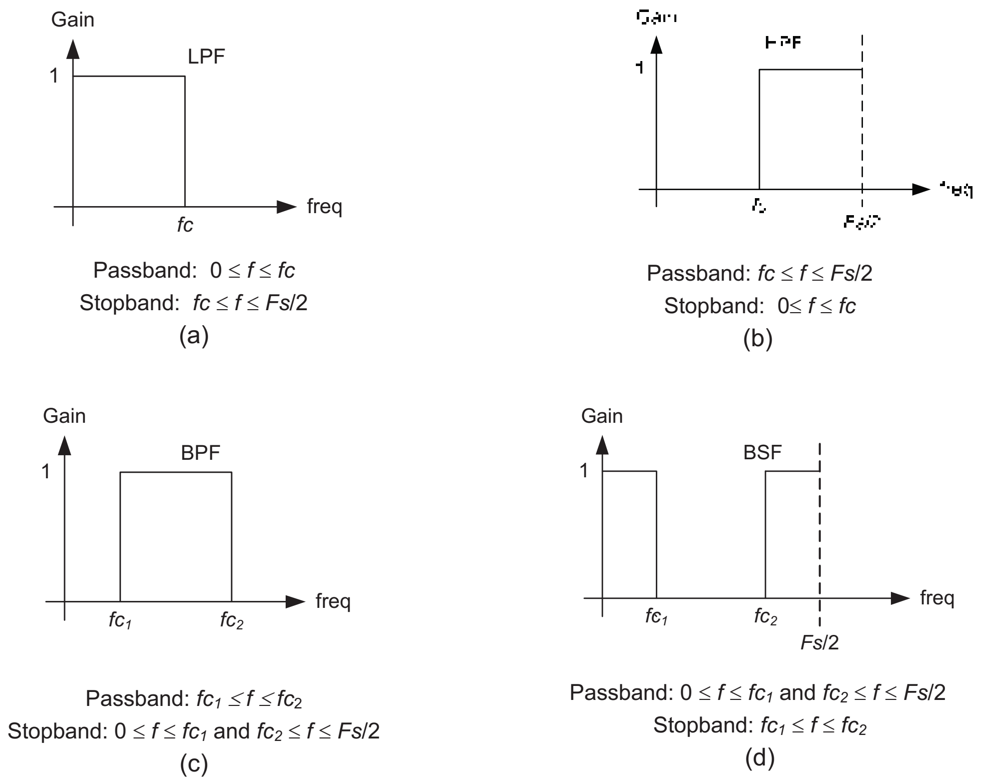

Gain
LPF
1
freq
fc
Passband: $0 \le f \le fc$
Stopband: $fc \le f \le Fs/2$
(a)
Gain
HPF
1
fc
Fs/2
freq
Passband: $fc \le f \le Fs/2$
Stopband: $0 \le f \le fc$
(b)
Gain
BPF
1
freq
fc_1
fc_2
Passband: $fc_1 \le f \le fc_2$
Stopband: $0 \le f \le fc_1$ and $fc_2 \le f \le Fs/2$
(c)
Gain
BSF
1
freq
fc_1
fc_2
Fs/2
Passband: $0 \le f \le fc_1$ and $fc_2 \le f \le Fs/2$
Stopband: $fc_1 \le f \le fc_2$
(d)

Figure 4.1: Ideal magnitude frequency responses (a) LPF (b) HPF (c) BPF (d) BSF.

4.1.1 Low-pass filter

A LPF passes all low-frequency components below the cut-off frequency, *fc* and blocks all higher frequency components above *fc*. Figure 4.2 shows the magnitude frequency response of a LPF in reality, where we can't design 'square' type of filters as shown in Figure 4.1. So, there needs to be transition band between the passband and stopband. The edge frequencies are the end frequencies of passband (*fp*) or stopband (*fs*). So, a practical LPF will allow frequency components below *fp* and remove components higher than *fs*.

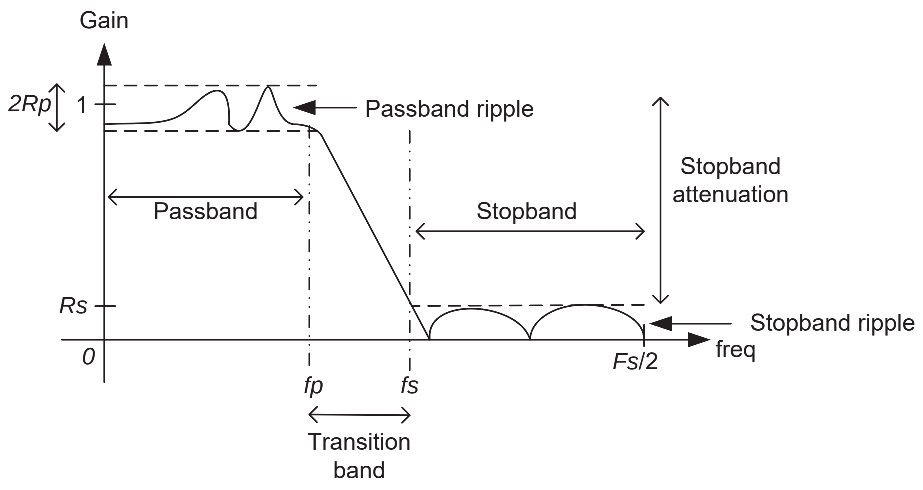


Figure 4.2: Magnitude frequency response of a LPF.

For example, consider a combination of three sinusoidal signals: 2 Hz, 5 Hz and 11 Hz as shown in Figure 4.3.

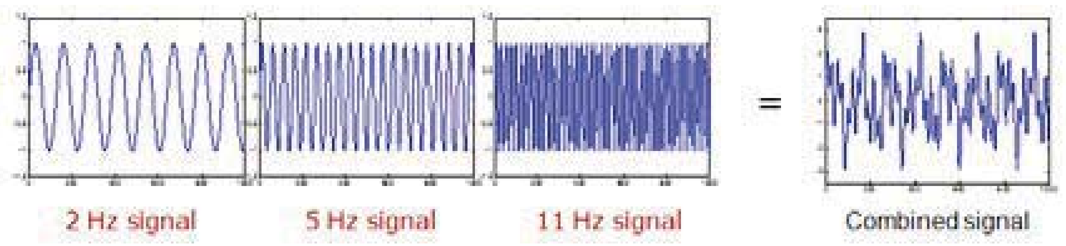


Figure 4.3: A combination of three sinusoidal signals.

The final output signals after LPF at *fp*=3 Hz with *fs*=4 Hz and *fp*=8 Hz with *fs*=9 Hz are shown in Figure 4.4.

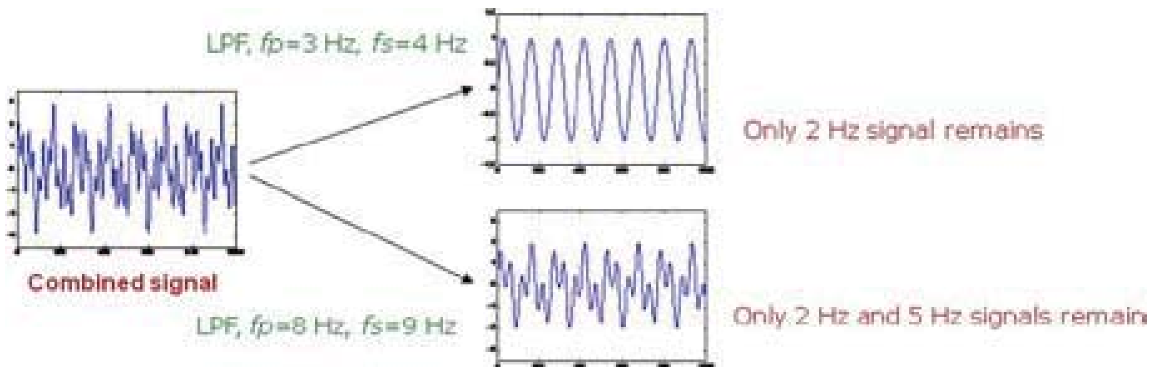


Figure 4.4: LPF of the three sinusoidal signals.

4.1.2 High-pass filter

HPF passes all high-frequency components above the cut-off frequency, *fc* and blocks all lower frequency components below *fc*. The magnitude frequency response of a HPF in reality is shown in Figure 4.5 where it allows frequency components higher than *fp* and remove components below *fs*.

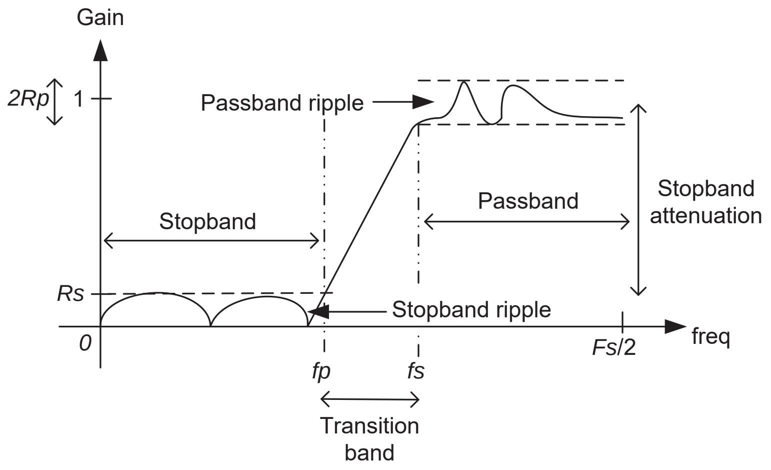


Figure 4.5: Magnitude frequency response of a HPF.

Consider the same combination of three sinusoidal signals: 2 Hz, 5 Hz and 11 Hz as previously. The final output signals after HPF at *fs*=3 Hz with *fp*=4 Hz and *fs*=8 Hz with *fp*=9 Hz are shown in Figure 4.6.

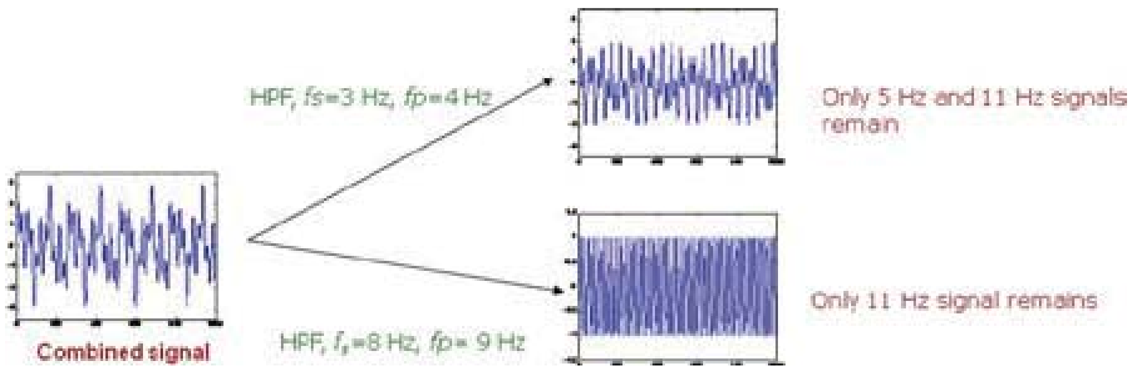


Figure 4.6: HPF of the three sinusoidal signals.

4.1.3 Band-pass and band-stop filters

BPF passes all frequency components between edge passband frequencies, fp_1<freq$_{(allow)}$<fp_2 and blocks all frequencies below and above edge stopband frequencies, freq$_{(block)}$<fs_1; freq$_{(block)}$>fs_2. A BPF can be designed using a LPF and HPF. BSF passes all frequency components lower and higher than edge passband frequencies, freq$_{(allow)}$<fp_1; freq$_{(allow)}$>fp_2 and blocks all frequencies between fs_1<freq$_{(block)}$<fs_2. The magnitude frequency responses of a BPF and a BSF are shown in Figures 4.7 and 4.8.

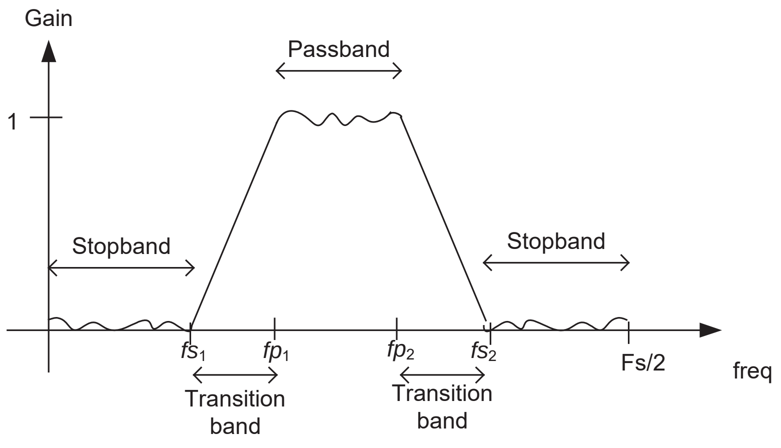


Figure 4.7: Magnitude frequency response of a BPF.

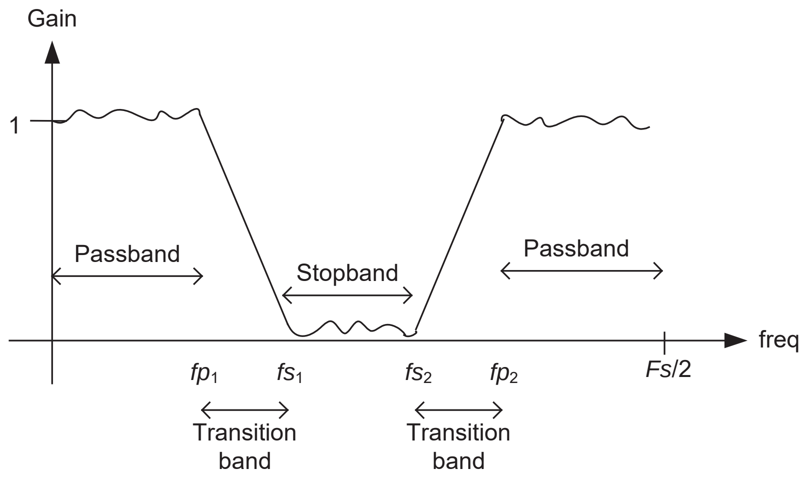
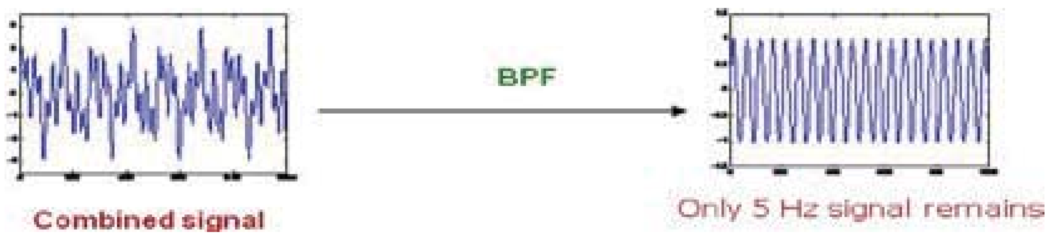
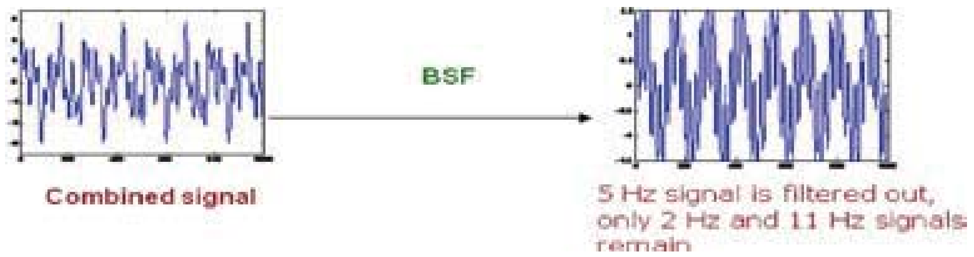


Figure 4.8: Magnitude frequency response of a BSF.

Figure 4.9 shows the output signals after applying BPF at fp_1=4 Hz, fp_2=6 Hz, fs_1=3 Hz, fs_2=7 Hz and BSF at fp_1=4 Hz, fp_2=6 Hz, fs_1=3 Hz, fs_2=7 Hz for the combination of the sinusoidal signals.



(a)



(b)

Figure 4.9: (a) BPF (b) BSF of the three sinusoidal signals.

4.2 Direct filtering in frequency domain

Filtering can be done directly in the frequency domain using the following steps:

- Obtain the Discrete Fourier Transform (DFT) of the signal (from 0 to *Fs*);
- Set to zero the values that are not in the required frequency range i.e. apply a rectangular window;
- Compute the Inverse Discrete Fourier Transform (IDFT).

For example, let use generate a combination of two sinusoidal signal with *f*1=8 Hz and *f*2=25 Hz with *N*=100, *Fs*=200 Hz (shown in Figure 4.10) and say, we wish to design a LPF with *fp*=10 Hz and *fs*=12 Hz. Compute `y=fft(x)` in MATLAB and apply the rectangular window, i.e. set the values `y(7:95)=0`. As MATLAB indexing starts from 1, `y(1:6)` represents DFT values from 0 to 10 Hz[16], which represents the passband range, the stopband range from 12 Hz to 100 Hz is represented by `y(7:51)`. Due to symmetry, we also need to create mirror images of the passband and stopband resulting in the rectangular window as shown in Figure 4.11 (a).

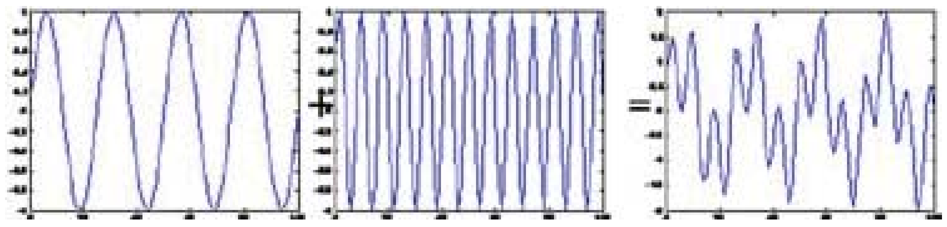


Figure 4.10: Combination of two sinusoidal signals (*f*1=8 Hz and *f*2=25 Hz).

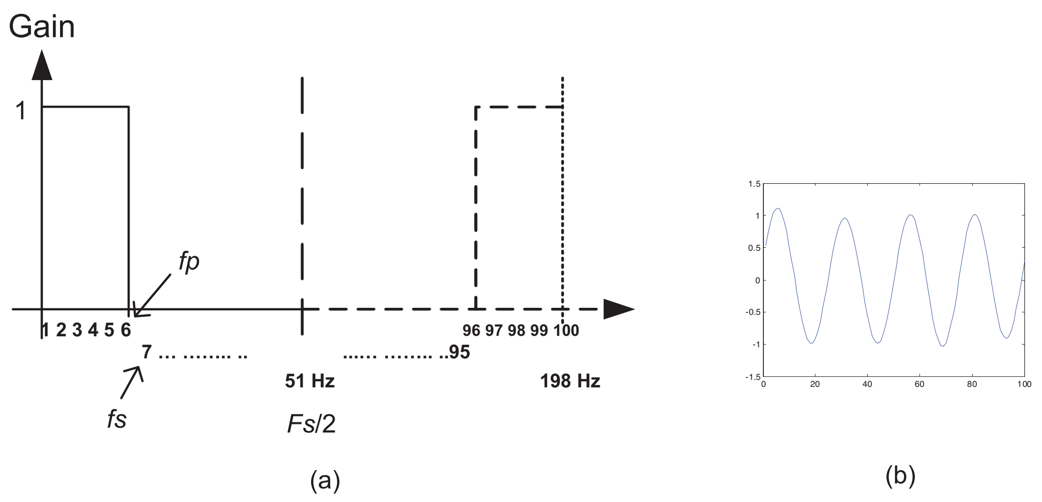


Figure 4.11: Direct LPF (a) rectangular window with *fp*=10 Hz and *fs*=12 Hz (b) filtered output.

Next, compute `yf=ifft(y,'symmetric')` and the low pass filtered signal is obtained as shown in Figure 4.11 (b). In MATLAB, it is useful to force conjugate symmetry, else complex values could be obtained due round-off errors in the `fft` and `ifft` operations. Figure 4.12 shows the whole procedure for the discussed example. This direct filtering method is simplistic to understand but has the disadvantage of high computation cost and requires chunks of data (i.e. real-time filtering is not possible). Thus we normally use finite impulse response (FIR) or infinite impulse response (IIR) filters to perform filtering.

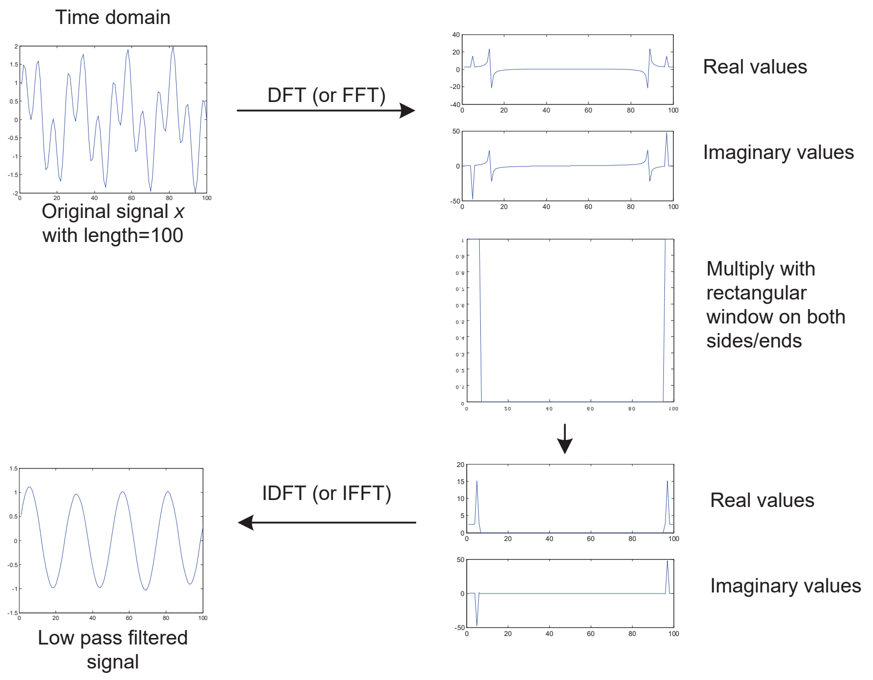


Figure 4.12: LPF example using direct filtering method.

4.3 Time domain filtering

To solve the problems of direct filtering, we could filter in time domain and there are several time domain filtering methods. We will look at design of simple FIR filters and IIR filters using MATLAB. The output from an IIR digital filter is made up of previous inputs and previous outputs:

$$y[n]=\sum_{k=1}^{M}B[k]x[n-k]+\sum_{j=1}^{N}A[j]y[n-j]. \tag{4.1}$$

where B and A are the filter coefficients. The output from a FIR digital filter is made up of previous inputs only, so there is no feedback:

$$y[n]=\sum_{k=1}^{M}B[k]x[n-k]. \tag{4.2}$$

Figure 4.13 shows an example comparing direct filtering in the frequency domain with time domain filtering. It should be obvious from this figure that filtering in time domain is computationally less complicated.