**B1.** Phổ của 1 tín hiệu sin là **phổ vạch (?)** hay không? Lý giải qua phần mềm và hình ảnh kết quả từ Matlab.

**B2.** Tạo 1 tín hiệu là tổng của 3 sóng sin có biên độ bằng nhau và có các tần số là 30, 50, 120 Hz. Thiết kế bộ lọc FIR (finite-duration impulse response) bằng phương pháp **cửa sổ lọc thành phần 50 Hz**. Nhận xét kết quả với các bộ lọc khác nhau.

- Đưa hình vẽ, code vào Word!

**Solution 1.** A continuous-time sinusoidal signal consists of a single spectral line (a single sine wave). However, a discrete-time sinusoidal signal may consist of more than one sine wave.

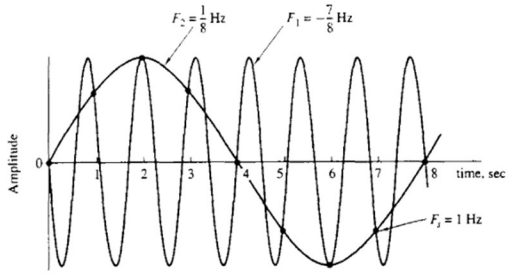
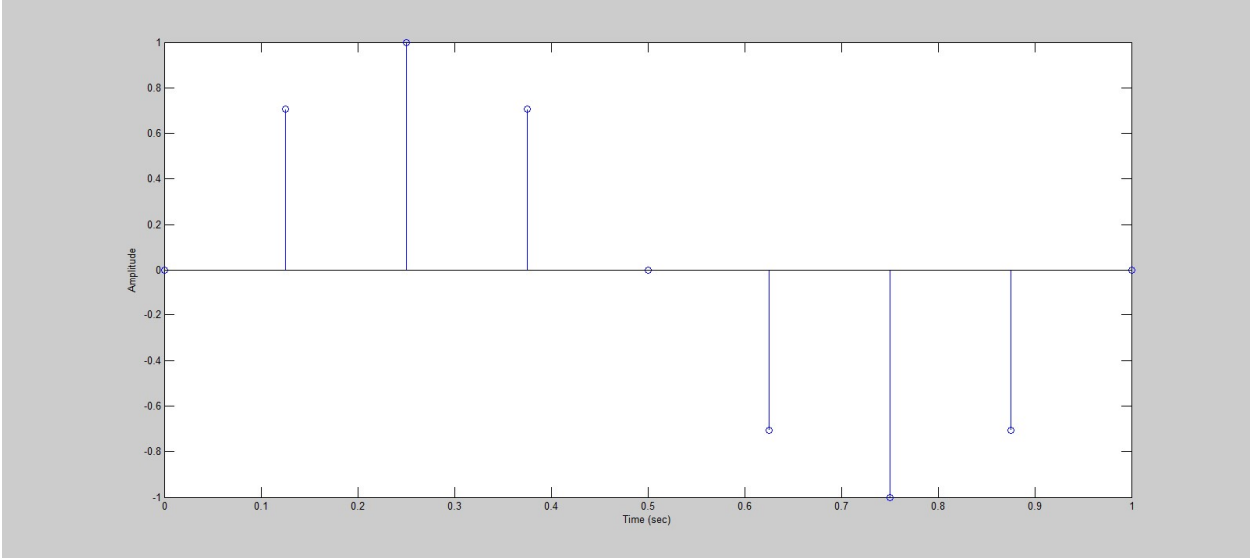



**Figure 1.18** Illustration of aliasing.

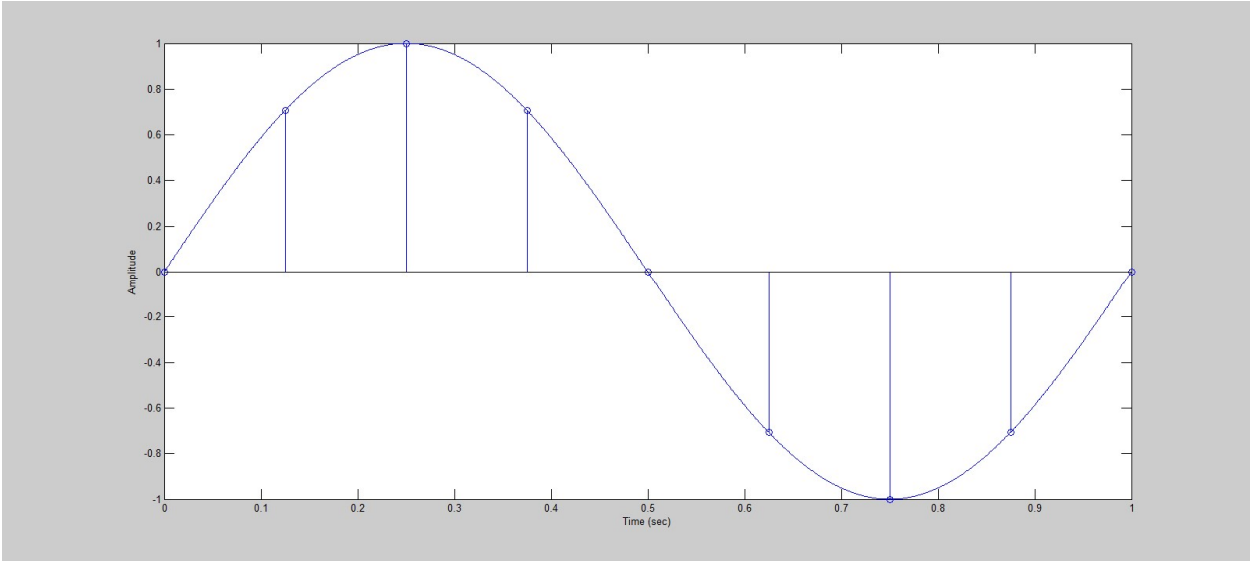
For example, let's plot a discrete-time sinusoidal signal at Fs = 8 Hz.

```
t = (0:8)*(1/8);              % Time vector

x = sin(2*pi*t);
figure, stem(t, x);
xlabel('Time (sec)');
ylabel('Amplitude');
```



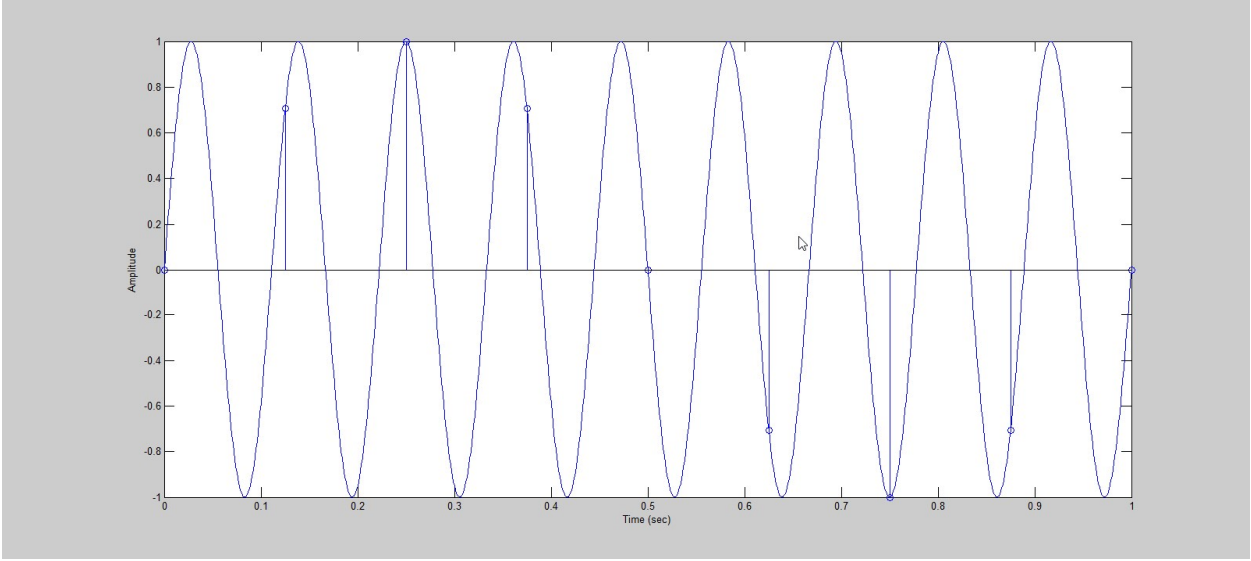
Let’s draw a 1 Hz sine wave to see that it could be one representation of the discrete-time signal:

```
t1 = (0:1000)*(1/1000);

x1 = sin(2*pi*1*t1);
figure, stem(t, x), hold on, plot(t1, x1);
xlabel('Time (sec)');
ylabel('Amplitude');
```



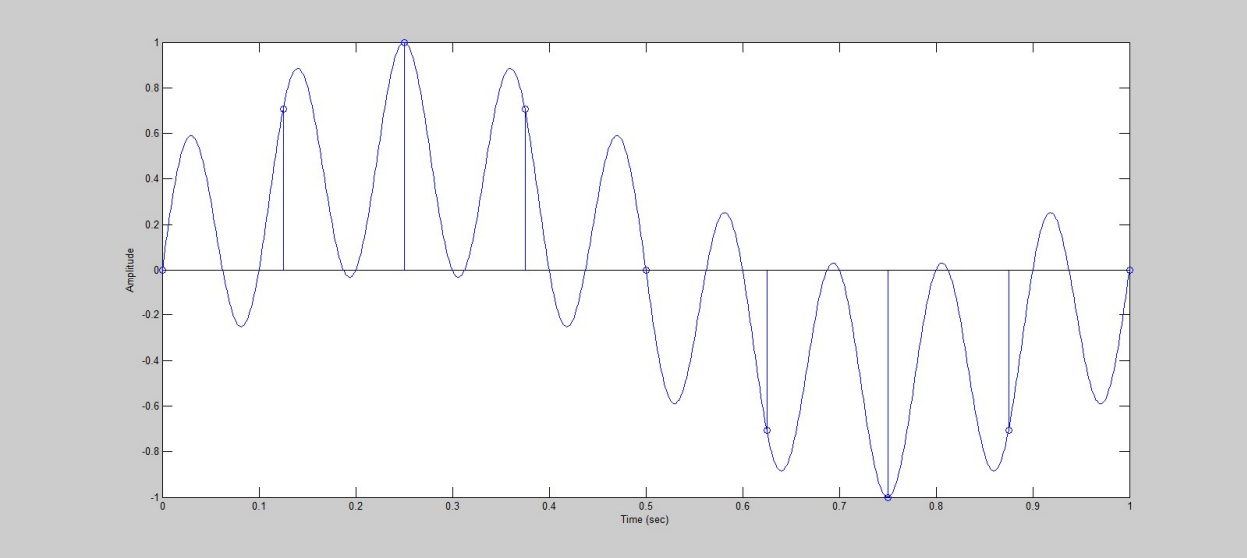

Now let’s draw a 9 Hz sine wave to see that it could also be another representation:

```
x2 = sin(2*pi*9*t1);
figure, stem(t, x), hold on, plot(t1, x2);
xlabel('Time (sec)');
ylabel('Amplitude');
```



Finally let's sum of the 2 sine waves (1 Hz and 9 Hz, each at half its amplitude):

```
x3 = 0.5*(x1+x2);
figure, stem(t, x), hold on, plot(t1, x3);
xlabel('Time (sec)');
ylabel('Amplitude');
```

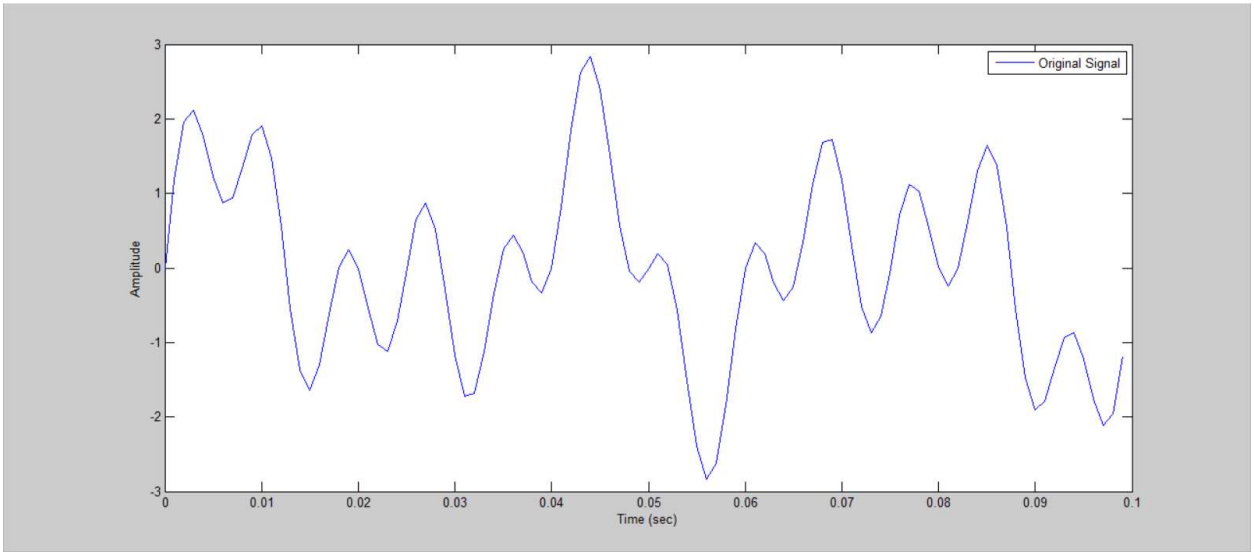



As we can see, there are many ways to add sine waves together to match a single discrete-time sinusoidal signal unless we limit the maximum frequency F (the highest frequency sine wave we can have). Thus, a discrete-time sinusoidal signal may consist of more than one sine wave or spectral line.

==========

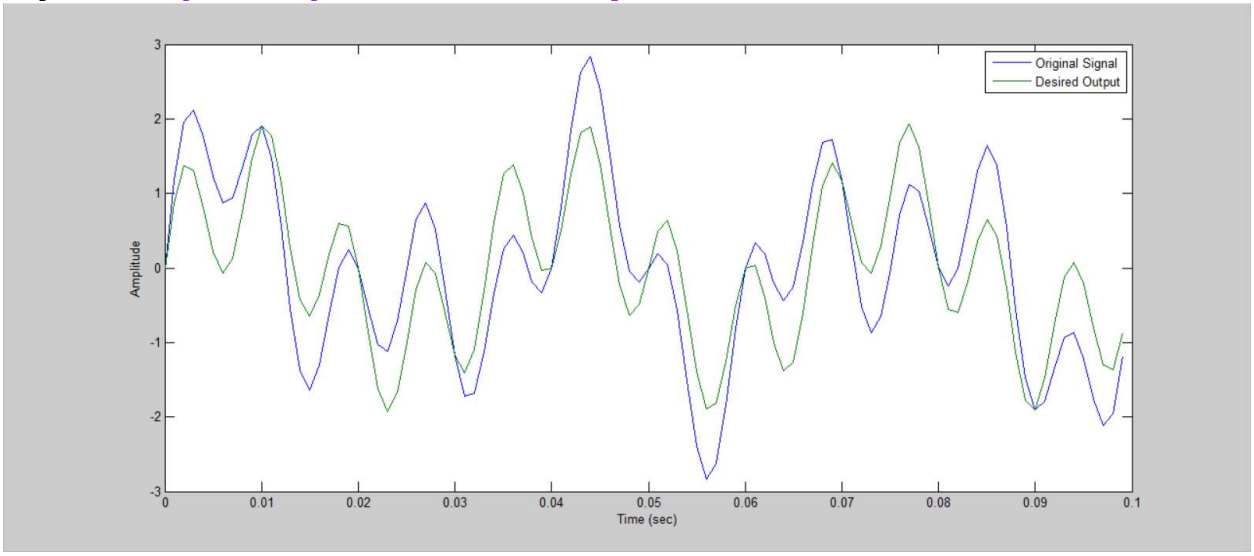
**Solution 2.** Firstly let's plot the given signal which is the sum of 3 sines:

```
Fs = 1000;                    % Sampling frequency
T = 1/Fs;                     % Sample time
L = 100;                      % Length of signal
t = (0:L-1)*T;                % Time vector

x1 = sin(2*pi*30*t);
x2 = sin(2*pi*50*t);
x3 = sin(2*pi*120*t);

x = x1 + x2 + x3;
figure, plot(t, x);
xlabel('Time (sec)');
ylabel('Amplitude');
legend('Original Signal');
```




We want to filter out the 50 Hz sine wave. Let's compare the target signal (50 Hz sine wave removed) with the original:

```
y = x1 + x3;
figure, plot(t, x, t, y);
xlabel('Time (sec)');
ylabel('Amplitude');
legend('Original Signal', 'Desired Output');
```

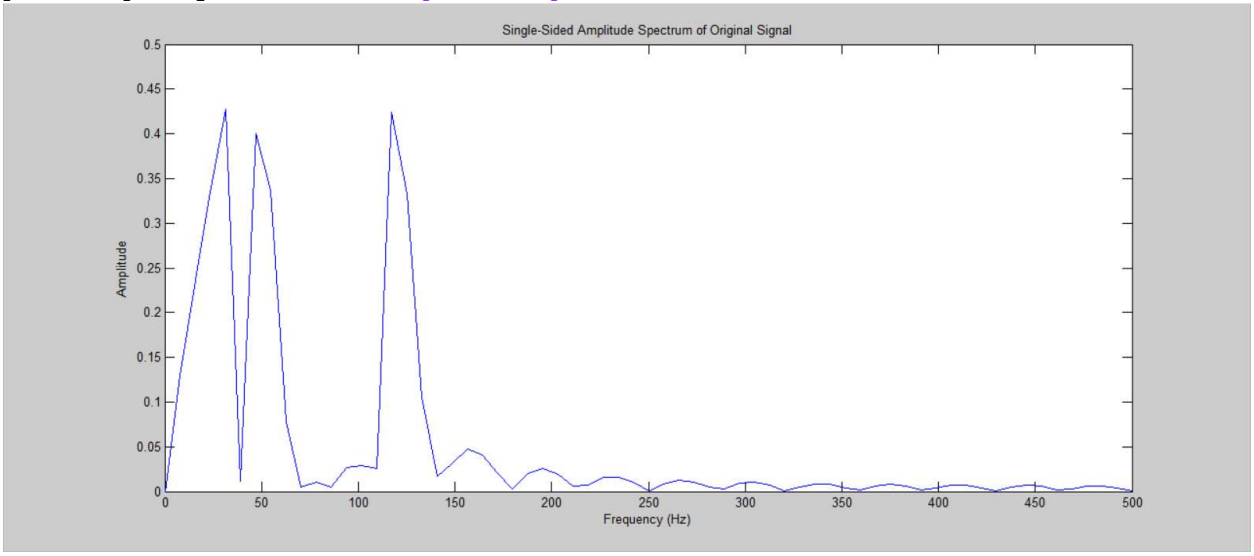



Looking at the time domain doesn't give us a good sense of comparison. Let's plot these on the frequency domain. I use a helper function to convert the signal to frequency domain and plot:

```
function plotFrequency(Fs, y, name)
    L = size(y, 2);
    NFFT = 2^nextpow2(L);
    F = linspace(0,Fs/2,NFFT/2+1);
    Y = fft(y,NFFT)/L;
    Y = abs(Y(1:NFFT/2+1));
    figure, plot(F, Y);
    ylim([0 0.5]);
    t = strcat('Single-Sided Amplitude Spectrum of ', name);
    title(t);
    xlabel('Frequency (Hz)');
    ylabel('Amplitude');
```

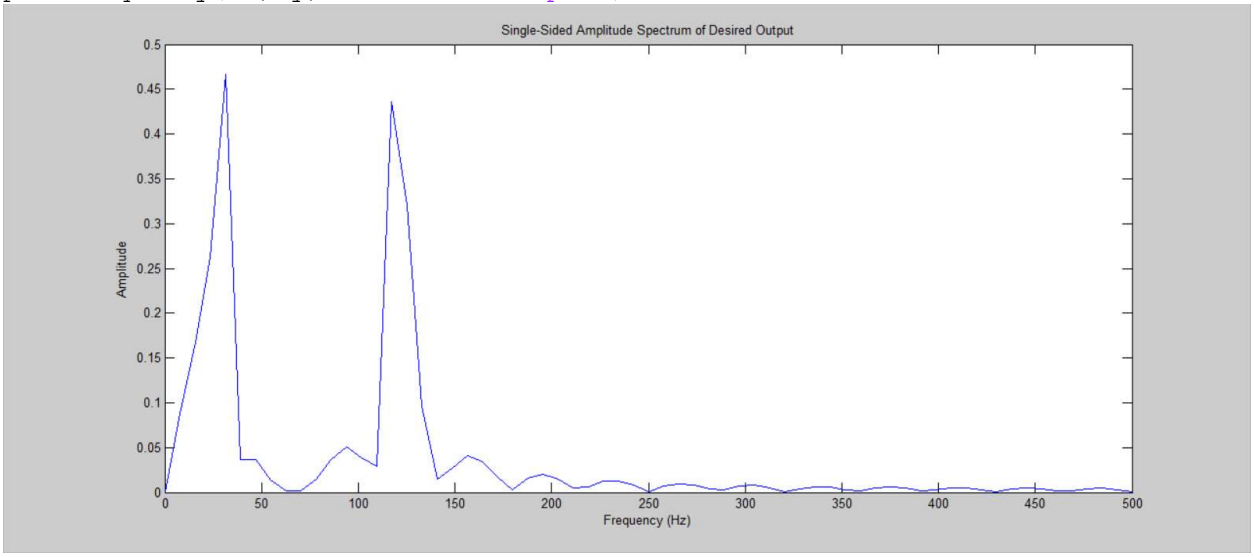
First the original signal:

```
plotFrequency(Fs, x, ' Original Signal');
```




We can see clearly the 3 peaks that represent our 3 sine waves. So the desired output should consist of only 2 peaks:

```
plotFrequency(Fs, y, ' Desired Output');
```



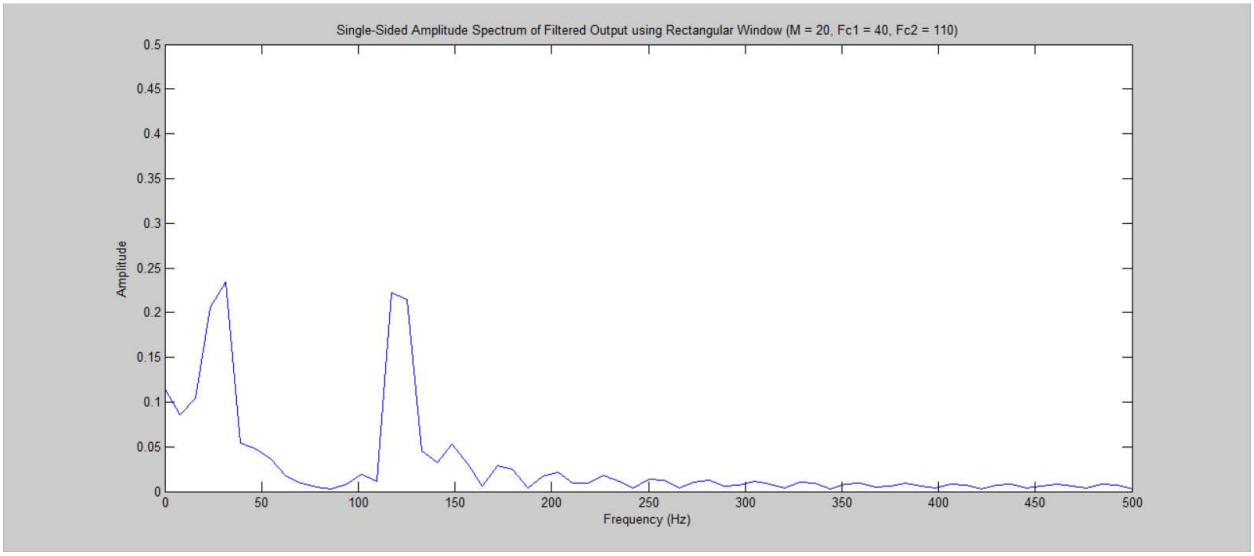
To filter out a specific frequency, we use a bandstop filter. Let the filter order M = 20 since our signal length is only 100.

First we try a rectangular window. The sampling frequency is 1000 Hz. Since we want to keep the frequency 30 Hz and 120 Hz and filter out 50 Hz, 30 Hz < Fc1 < 50 Hz and 50 Hz < Fc2 < 120 Hz. Let's try Fc1 = 40 Hz and Fc2 = 110 Hz. I create a small convenient function:

```
function Hd = Hd_rectangular(Fs, N, Fc1, Fc2)
    flag = 'scale';  % Sampling Flag
    win = rectwin(N+1);
    b  = fir1(N, [Fc1 Fc2]/(Fs/2), 'stop', win, flag);
    Hd = dfilt.dffir(b);
```

Let's apply the filter and see the result:

```
Hd1 = Hd_rectangular(Fs, 20, 40, 110);
y1 = filter(Hd1, x);
plotFrequency(Fs, y1, ' Filtered Output using Rectangular Window (M = 20, Fc1 = 40, Fc2 = 110)');
```

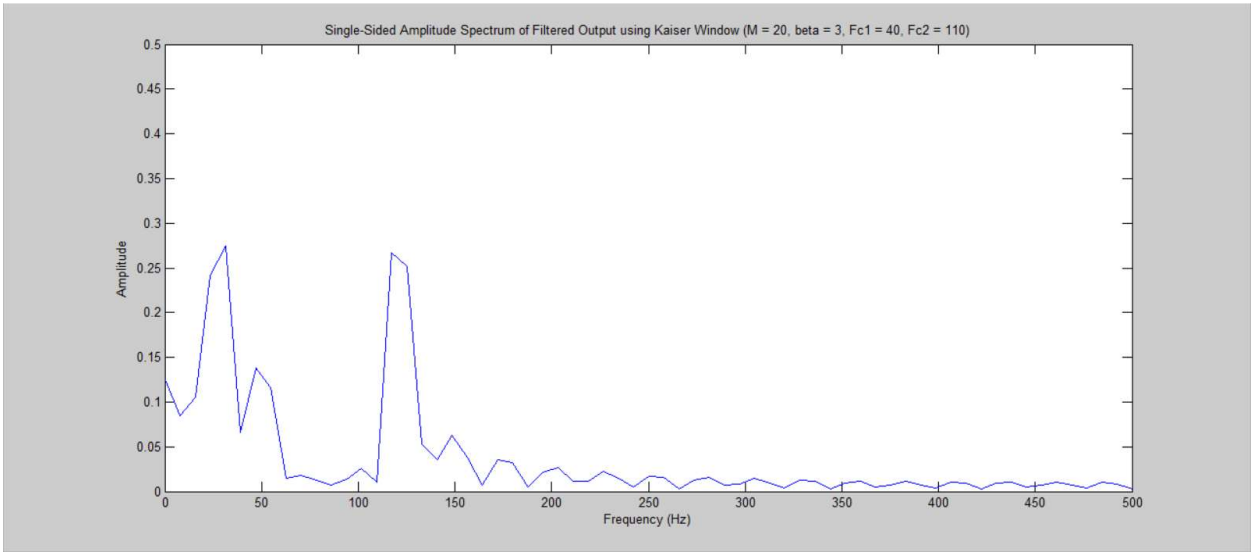



We can try using a different window. Let's try the Kaiser window with beta = 3 and keep the other params the same. Again I made a convenient function:

```
function Hd = Hd_kaiser(Fs, N, Fc1, Fc2, Beta)
    flag = 'scale';  % Sampling Flag
    win = kaiser(N+1, Beta);
    b  = fir1(N, [Fc1 Fc2]/(Fs/2), 'stop', win, flag);
    Hd = dfilt.dffir(b);
```

Let's see and compare the result:

```
Hd2 = Hd_kaiser(Fs, 20, 40, 110, 3);
y2 = filter(Hd2, x);
plotFrequency(Fs, y2, ' Filtered Output using Kaiser Window (M = 20, beta = 3, Fc1 = 40, Fc2 = 110)');
```

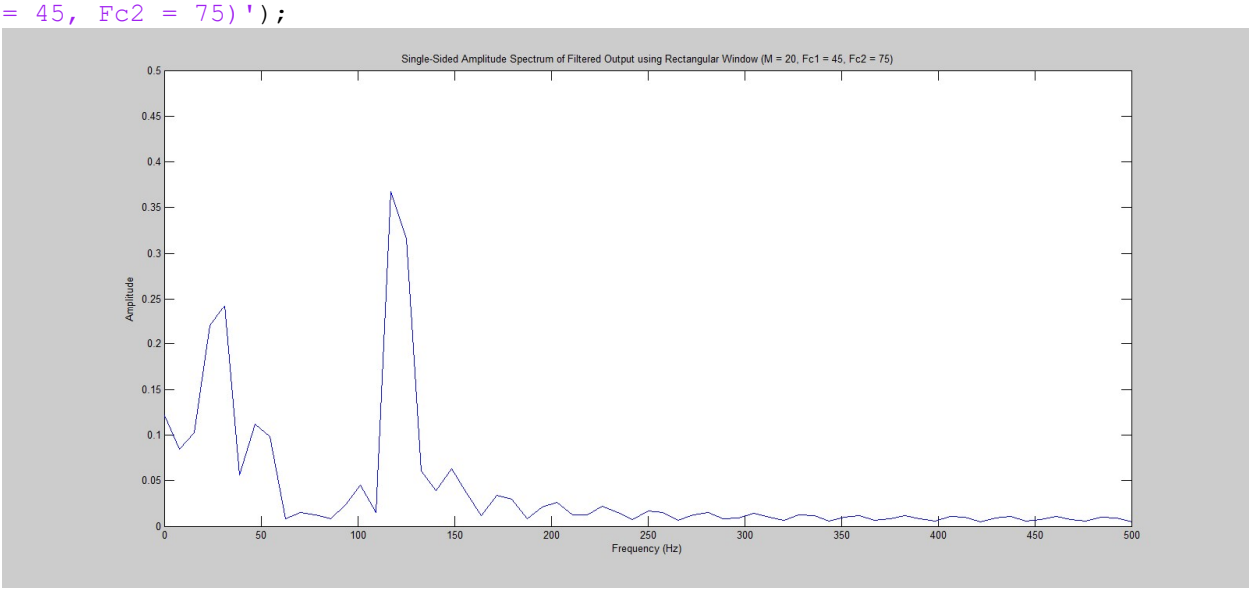



The Kaiser window seems to be less effective than the rectangular window at removing 50 Hz. However it preserves the amplitude of the other frequencies (30 Hz and 120 Hz) slightly better.

The result could also be largely affected by our choice of cutoff frequencies Fc1 and Fc2. Let's try a closer range with Fc1 = 45 Hz and Fc2 = 75 Hz.

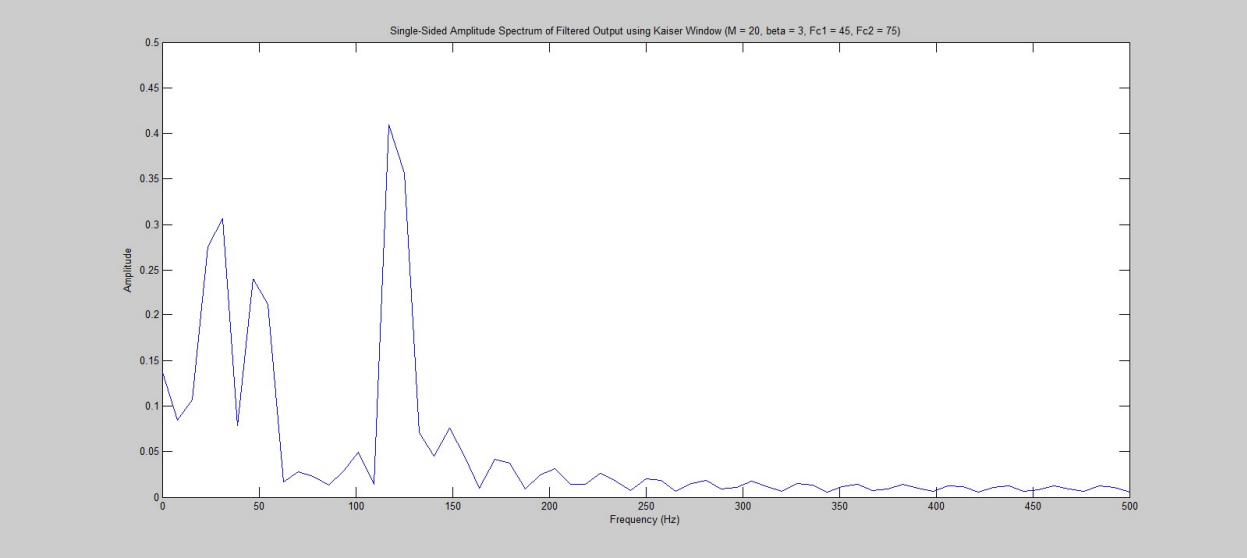
First is the rectangular window:

```
Hd3 = Hd_rectangular(Fs, 20, 45, 75);
y3 = filter(Hd3, x);
plotFrequency(Fs, y3, ' Filtered Output using Rectangular Window (M = 20, Fc1 = 45, Fc2 = 75)');
```



Then for the Kaiser window:

```
Hd4 = Hd_kaiser(Fs, 20, 45, 75, 3);
y4 = filter(Hd4, x);
plotFrequency(Fs, y4, ' Filtered Output using Kaiser Window (M = 20, beta = 3, Fc1 = 45, Fc2 = 75)');
```




As we can see, when the cutoff frequencies are closer to 50 Hz, or further from 30 Hz and 120 Hz, the filter affects the other frequencies less (especially for the 120 Hz frequency as we decrease Fc2 from 110 to 75). But the tradeoff is that the 50 Hz frequency is now stronger. Again, the filter with Kaiser window leaves a higher 50 Hz peak than rectangular window, but also higher 30 Hz and 120 Hz peaks.

All Matlab code:

B1.m:

```
t = (0:8)*(1/8);              % Time vector

x = sin(2*pi*t);
figure, stem(t, x);
xlabel('Time (sec)');
ylabel('Amplitude');

t1 = (0:1000)*(1/1000);

x1 = sin(2*pi*1*t1);
figure, stem(t, x), hold on, plot(t1, x1);
xlabel('Time (sec)');
ylabel('Amplitude');

x2 = sin(2*pi*9*t1);
figure, stem(t, x), hold on, plot(t1, x2);
xlabel('Time (sec)');
ylabel('Amplitude');
```

```
x3 = 0.5*(x1+x2);
figure, stem(t, x), hold on, plot(t1, x3);
xlabel('Time (sec)');
ylabel('Amplitude');
```

B2.m:

```
Fs = 1000;                  % Sampling frequency
T = 1/Fs;                   % Sample time
L = 100;                    % Length of signal
t = (0:L-1)*T;              % Time vector

x1 = sin(2*pi*30*t);
x2 = sin(2*pi*50*t);
x3 = sin(2*pi*120*t);

x = x1 + x2 + x3;
figure, plot(t, x);
xlabel('Time (sec)');
ylabel('Amplitude');
legend('Original Signal');

y = x1 + x3;
figure, plot(t, x, t, y);
xlabel('Time (sec)');
ylabel('Amplitude');
legend('Original Signal', 'Desired Output');

plotFrequency(Fs, x, ' Original Signal');

plotFrequency(Fs, y, ' Desired Output');

Hd1 = Hd_rectangular(Fs, 20, 40, 110);
y1 = filter(Hd1, x);
plotFrequency(Fs, y1, ' Filtered Output using Rectangular Window (M = 20, Fc1
= 40, Fc2 = 110)');

Hd2 = Hd_kaiser(Fs, 20, 40, 110, 3);
y2 = filter(Hd2, x);
plotFrequency(Fs, y2, ' Filtered Output using Kaiser Window (M = 20, beta =
3, Fc1 = 40, Fc2 = 110)');

Hd3 = Hd_rectangular(Fs, 20, 45, 75);
y3 = filter(Hd3, x);
plotFrequency(Fs, y3, ' Filtered Output using Rectangular Window (M = 20, Fc1
= 45, Fc2 = 75)');

Hd4 = Hd_kaiser(Fs, 20, 45, 75, 3);
y4 = filter(Hd4, x);
plotFrequency(Fs, y4, ' Filtered Output using Kaiser Window (M = 20, beta =
3, Fc1 = 45, Fc2 = 75)');
```

Hd_rectangular.m:

```
function Hd = Hd_rectangular(Fs, N, Fc1, Fc2)
    flag = 'scale';  % Sampling Flag
    win = rectwin(N+1);
    b  = fir1(N, [Fc1 Fc2]/(Fs/2), 'stop', win, flag);
    Hd = dfilt.dffir(b);
```

Hd_kaiser.m:

```
function Hd = Hd_kaiser(Fs, N, Fc1, Fc2, Beta)
    flag = 'scale';  % Sampling Flag
    win = kaiser(N+1, Beta);
    b  = fir1(N, [Fc1 Fc2]/(Fs/2), 'stop', win, flag);
    Hd = dfilt.dffir(b);
```

plotFrequency.m:

```
function plotFrequency(Fs, y, name)
    L = size(y, 2);
    NFFT = 2^nextpow2(L);
    F = linspace(0,Fs/2,NFFT/2+1);
    Y = fft(y,NFFT)/L;
    Y = abs(Y(1:NFFT/2+1));
    figure, plot(F, Y);
    ylim([0 0.5]);
    t = strcat('Single-Sided Amplitude Spectrum of ', name);
    title(t);
    xlabel('Frequency (Hz)');
    ylabel('Amplitude');
```