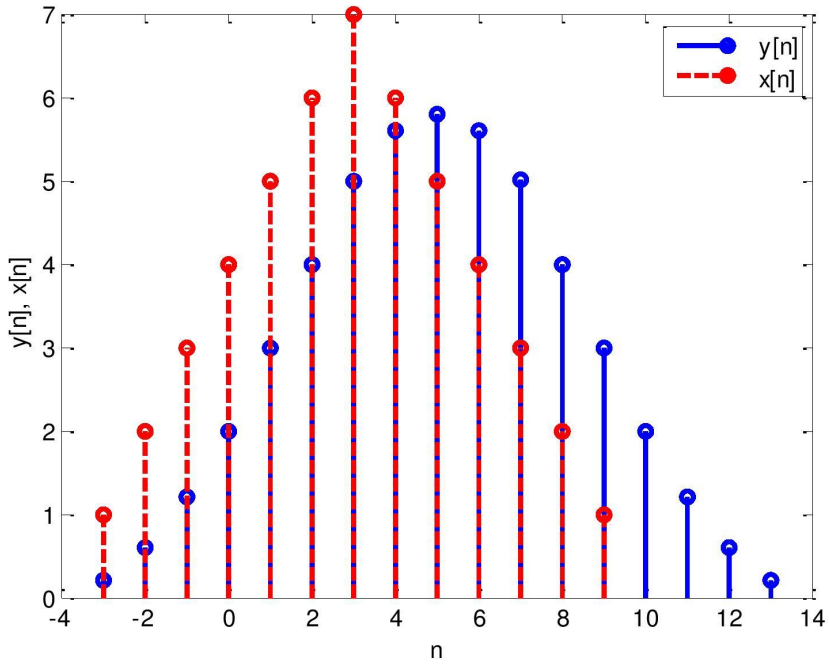
25)

a)

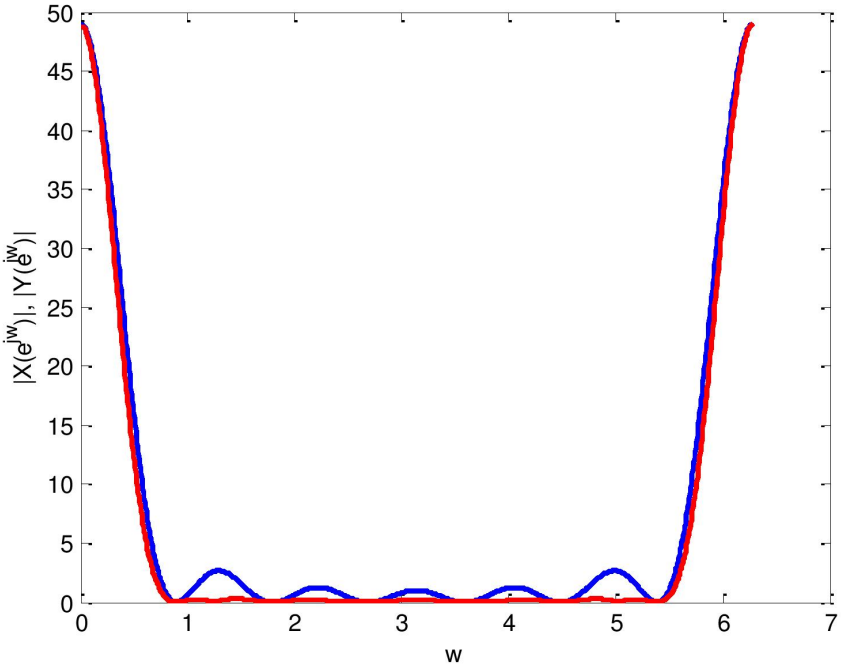
```
clear all, close all;
x = [1 2 3 4 5 6 7 6 5 4 3 2 1];
h = 1/5*[1 1 1 1 1];
y = conv(h,x);
n = -3:13;
figure, stem(n, y), xlabel('n'), ylabel('y[n], x[n]');
hold on
stem(-3:9, x, 'r--')
```

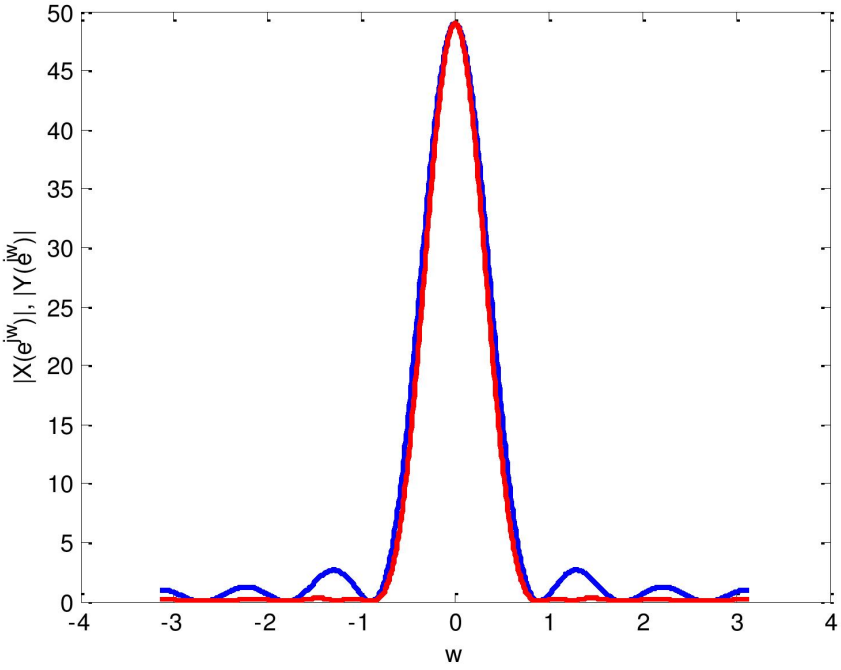



The sharp transitions in the input signal are smoothed since each output value is obtained by taking the average of two past and two future values in addition to the current one. This is clearly observed for the indices around the center point, where the input follows a triangular shape and the corresponding region follows a more bell-like shape for the output signal.

b)

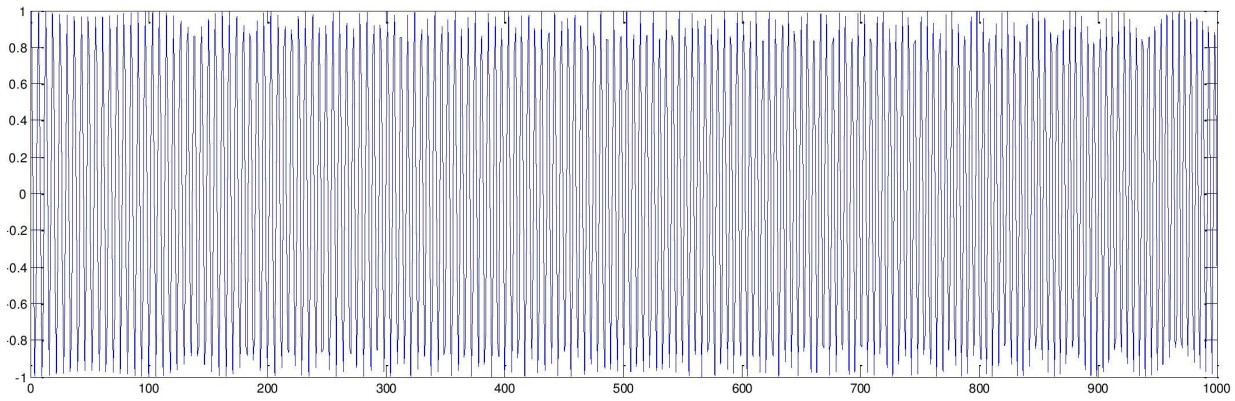
```
X_mag = abs(fft(x,1024));
Y_mag = abs(fft(y,1024));
w = 0:1023;
w = 2 * pi * w / 1024;
figure, plot(w, X_mag);
hold
plot(w, Y_mag,'r'), xlabel('w'), ylabel('|X(e^{jw})|, |Y(e^{jw})|');
w = -512:511;
w = pi * w / 512;
figure, plot(w, fftshift(X_mag));
hold
plot(w, fftshift(Y_mag),'r'),xlabel('w'),ylabel('|X(e^{jw})|, |Y(e^{jw})|');
set(findobj('type','line'),'linewidth',2)
```



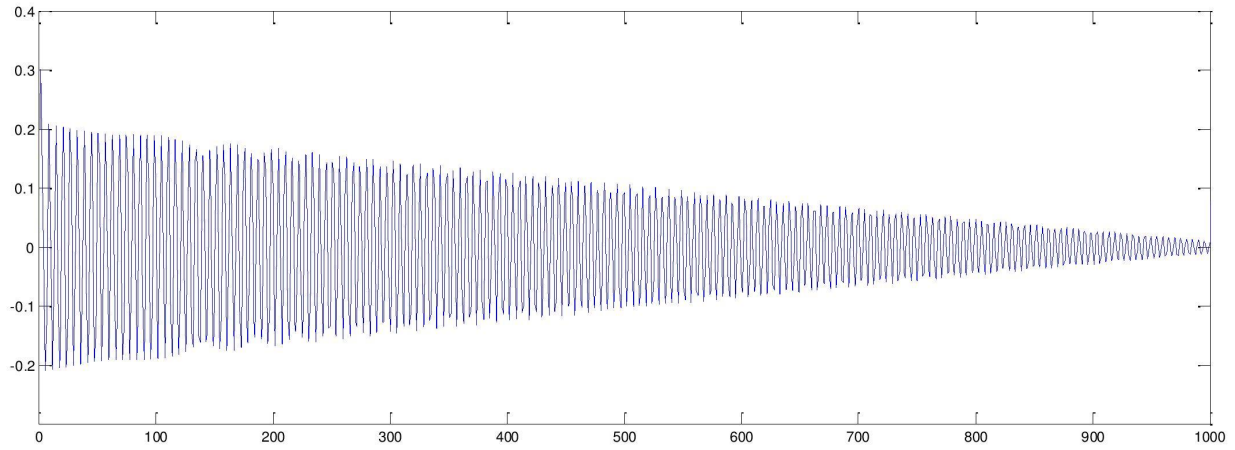




The high-frequency components are suppressed at the output of the filter (shown in red), whereas the low-frequency region is relatively unaffected by it.

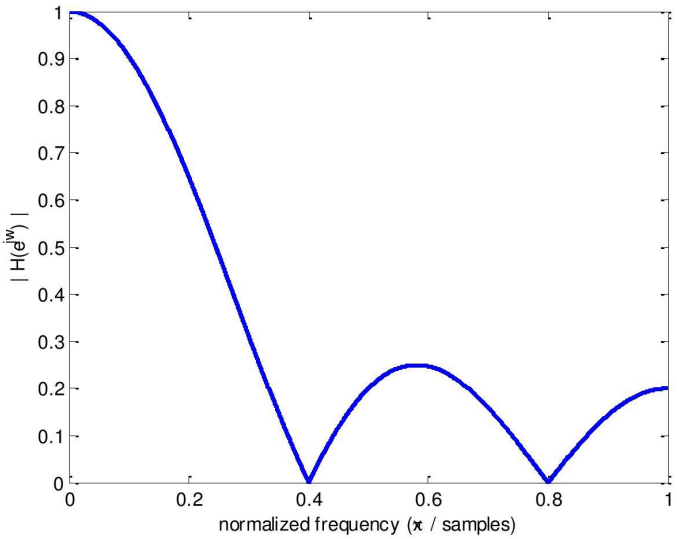
26) a,b,c,d,e,f)




x[n]



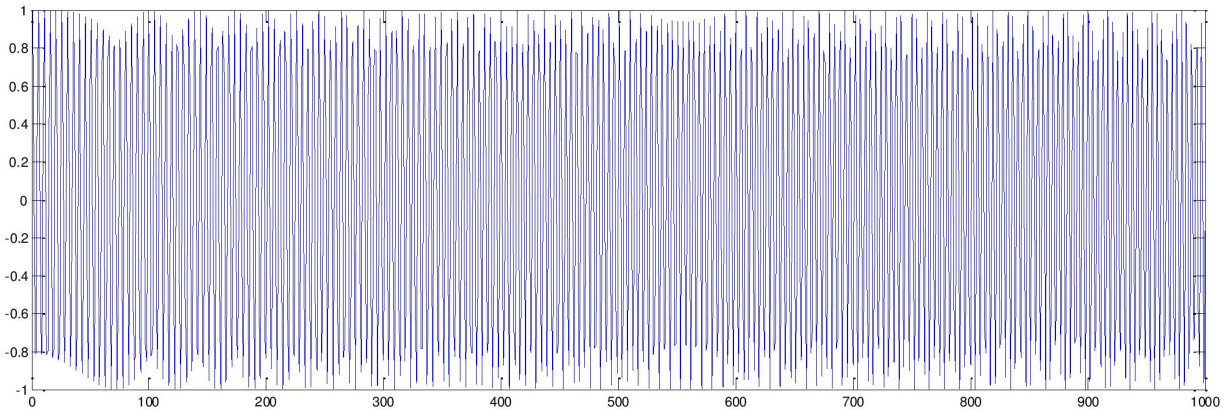

y[n]



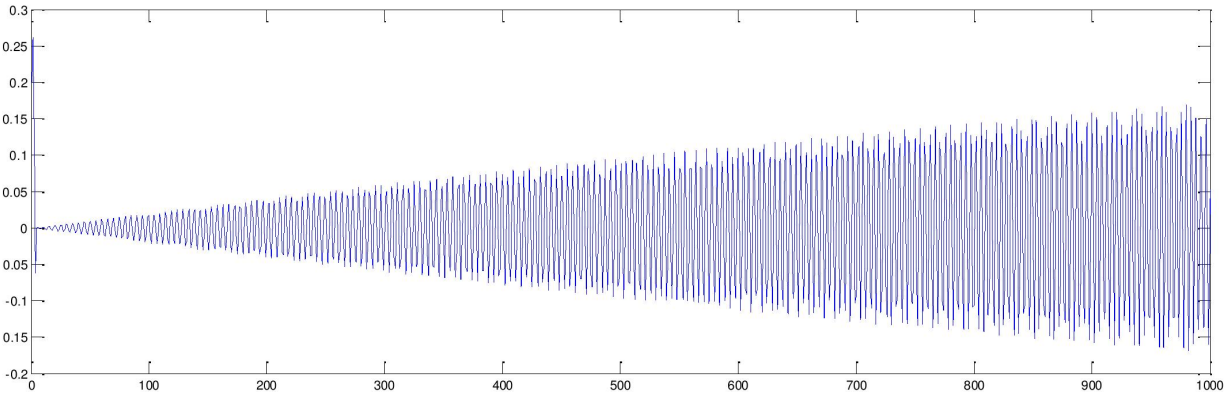

Magnitude response for the moving average filter

Since the filter gain decreases as we move from w0 = 0.33π (initial frequency of chirp signal) to 0.363π (final frequency), the output signal fades at the end of the signaling duration, where the instantaneous frequency is high.

e)

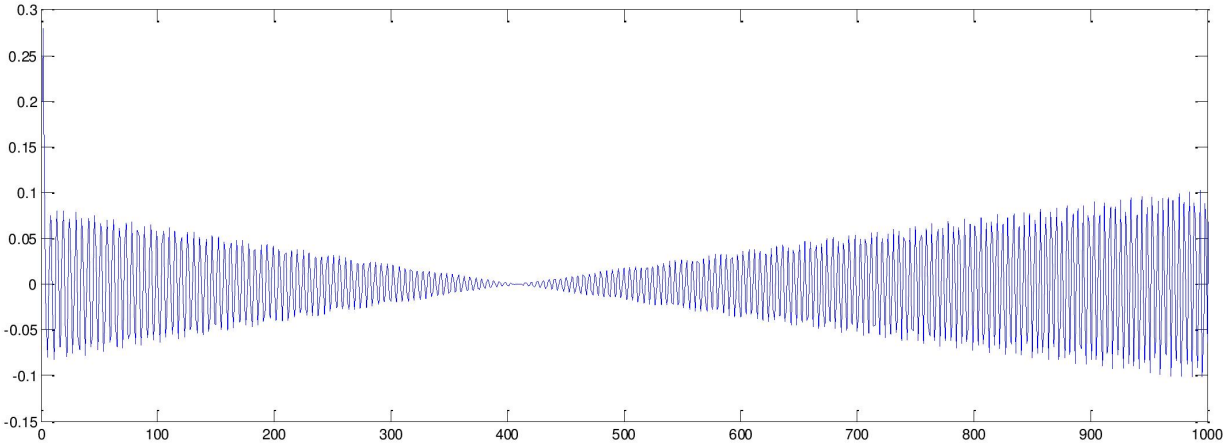



x[n] with w0 = 0.4π




y[n] with w0 = 0.4π

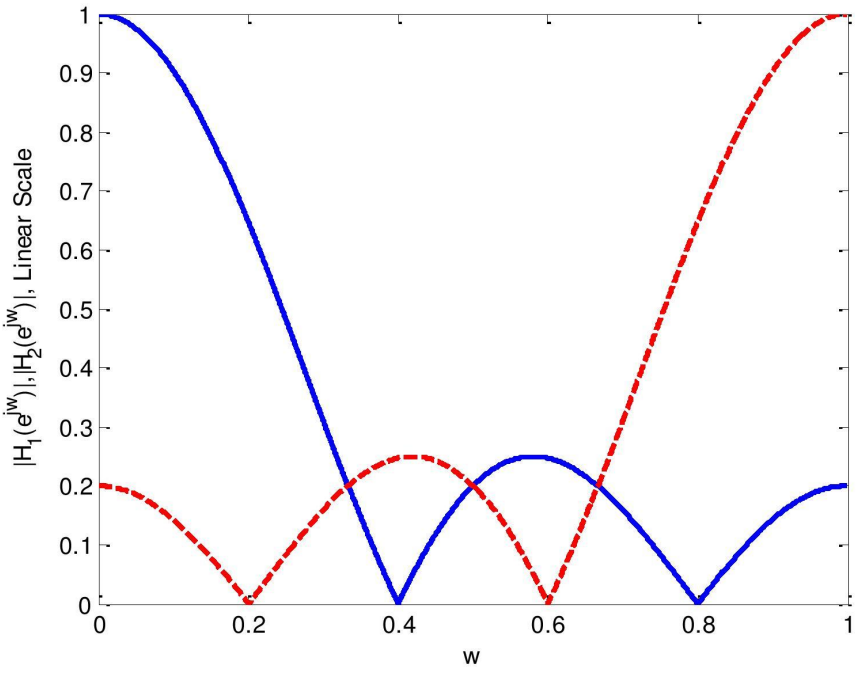
The filter has a null at 0.4 π and hence the output signal is zero at the beginning of the signaling duration. As the frequency of the input chirp signal increases with time, the output signal gets stronger since the magnitude response of the filter takes larger values for frequency increasing from 0.4 π to 0.44 π. One may get following output signal plot if the starting frequency is set as 0.37 π.




y[n] with w0 = 0.37π

27)

```
clear all
close all
filt_length = 5;
h1 = ones(1,filt_length) / filt_length;
h2 = [1 -1 1 -1 1] / filt_length;
N= 1000;
[H1,W] = freqz(h1,1,N);
figure
plot(W/pi,abs(H1)), xlabel('w');
[H2,W] = freqz(h2,1,N);
hold on
plot(W/pi,abs(H2),'r--'), xlabel('w');
ylabel('|H_1(e^{jw})|,|H_2(e^{jw})|, Linear Scale');
set(findobj('type','line'),'linewidth',2)
```




The second filter (dotted red line) is basically a high-pass filter as opposed to the low-pass characteristics of the filter of item-3, which is drawn in solid blue color. However, one should be careful in deciding in these characteristics, because, as an example, it is easy to see that a sinusoidal component at w=0.4π, is completely eliminated at the output of the “low-pass” filter while it is only weakened four times (amplitude-wise) when input to the “high-pass” filter.

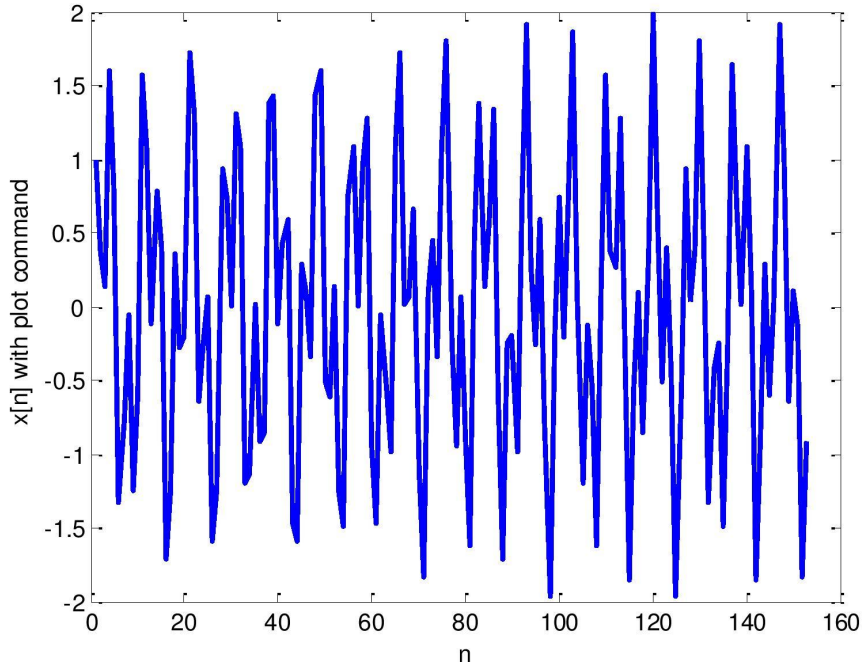
28 )

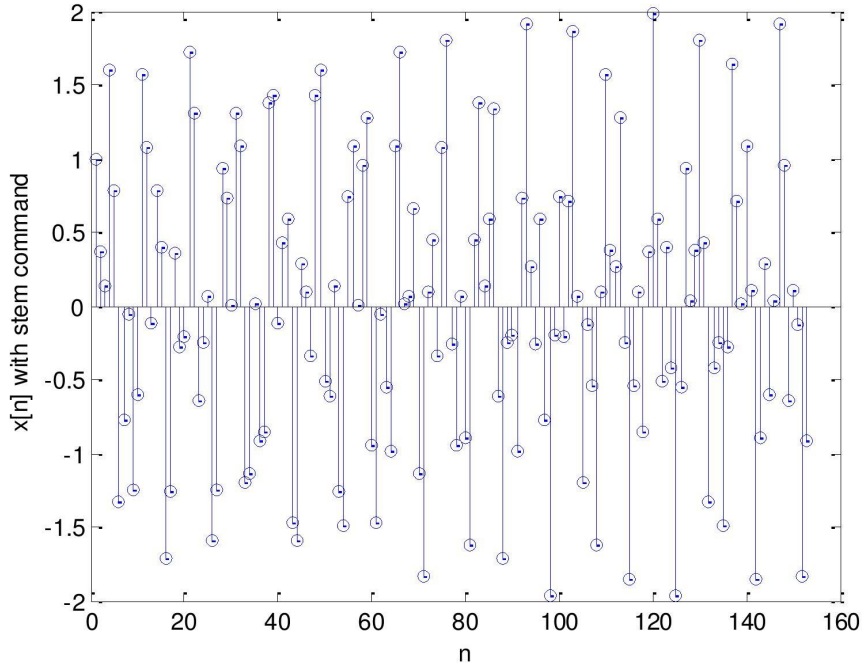
a) The fundamental periods are 9 and 17 respectively for the sine and the cosine components.

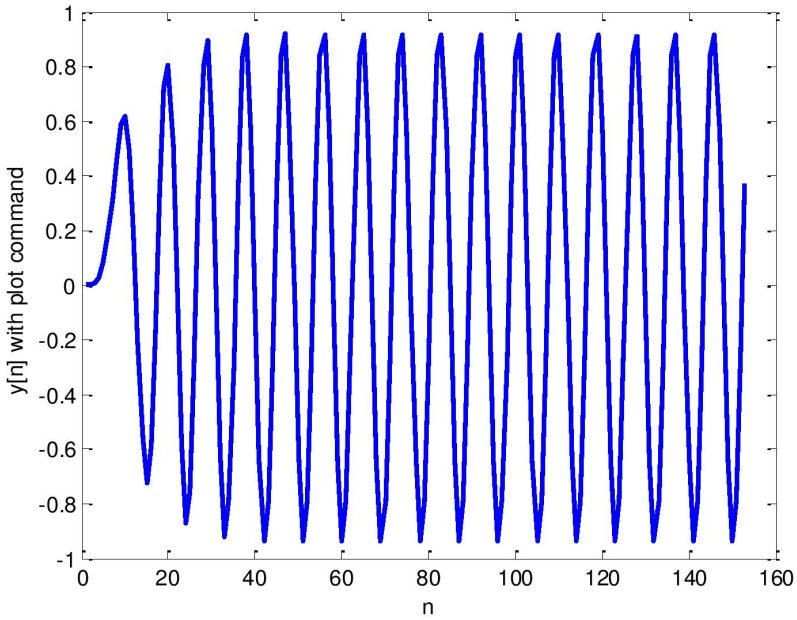
b) The butter function accepts its cut-off frequency argument $f_c$ normalized by $\pi$ and designs the filter within $[-\pi, \pi]$ radial frequency region, which corresponds to the normalized region [-1,+1]. Hence for the cut-off parameter $f_c$, the low-pass filter designed resides in $[-f_c \pi, f_c \pi]$.

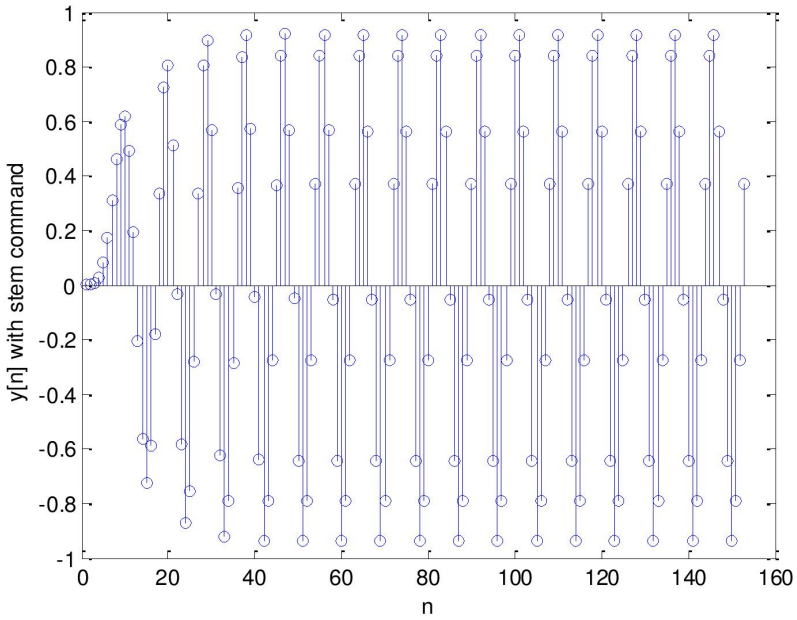
c) The filter accepts the filter in terms of its parameters defining the rational transfer function. On the other hand, conv function operates on the input signal by using the impulse response of the filter.

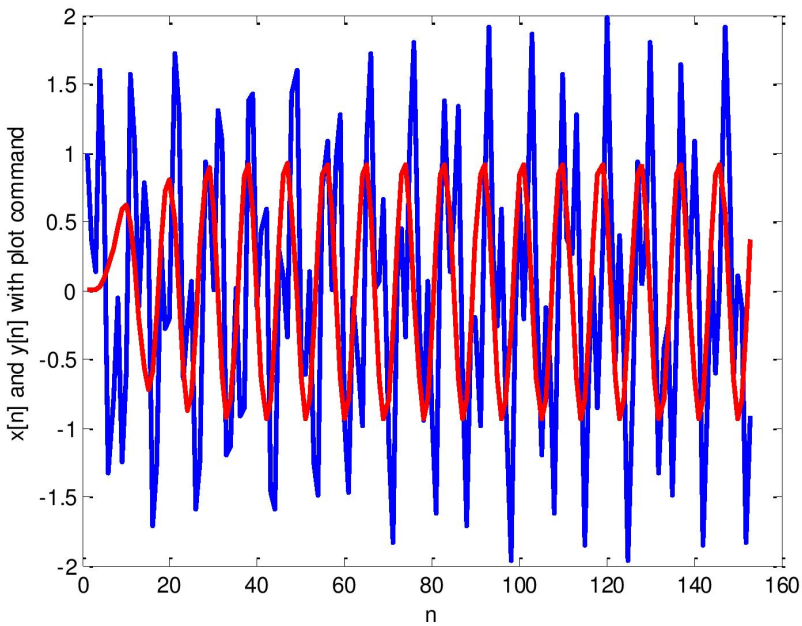
d) e) f)

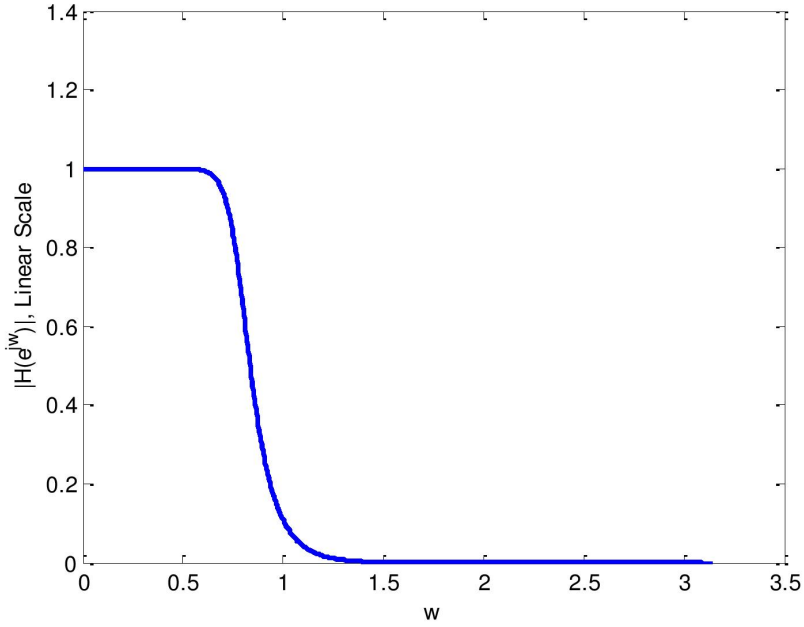


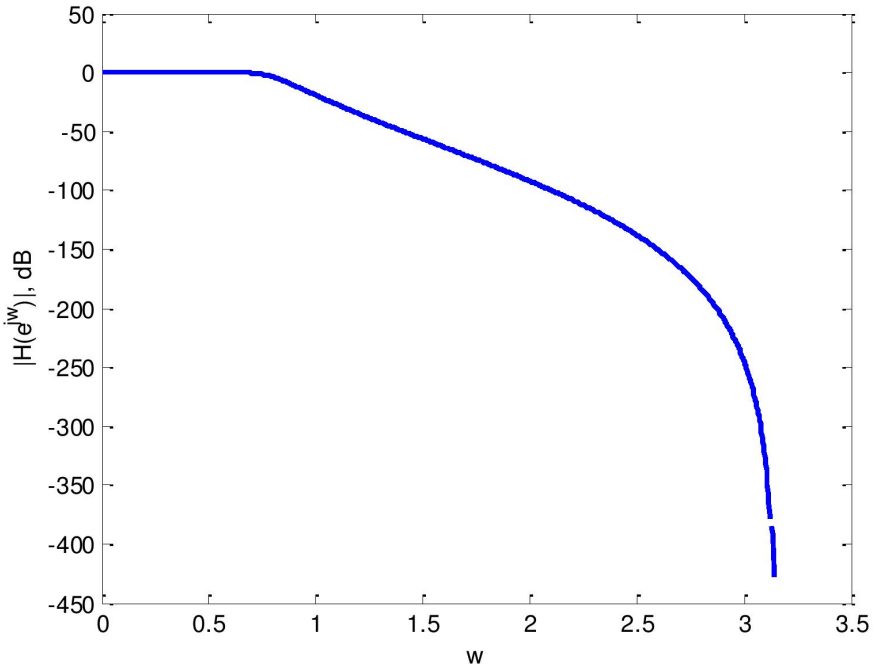




y[n] with plot command
n


y[n] with stem command
n


x[n] and y[n] with plot command
n

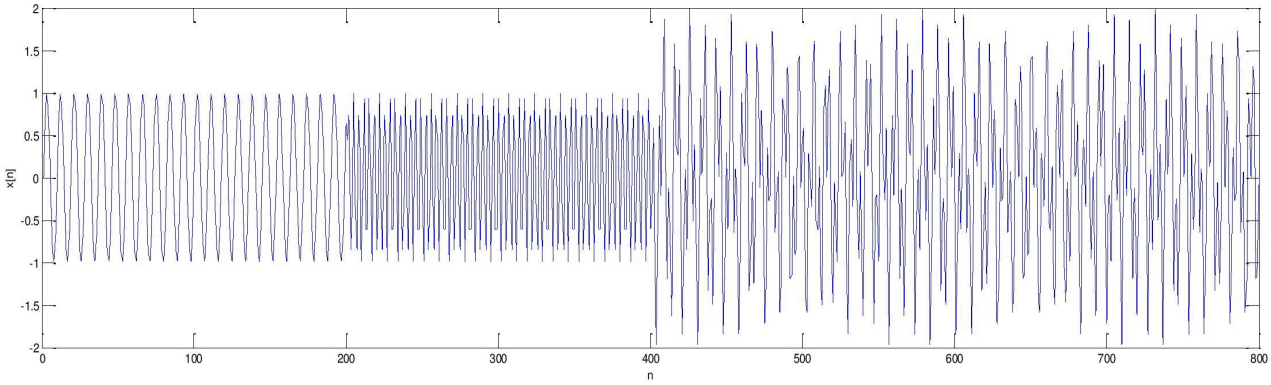


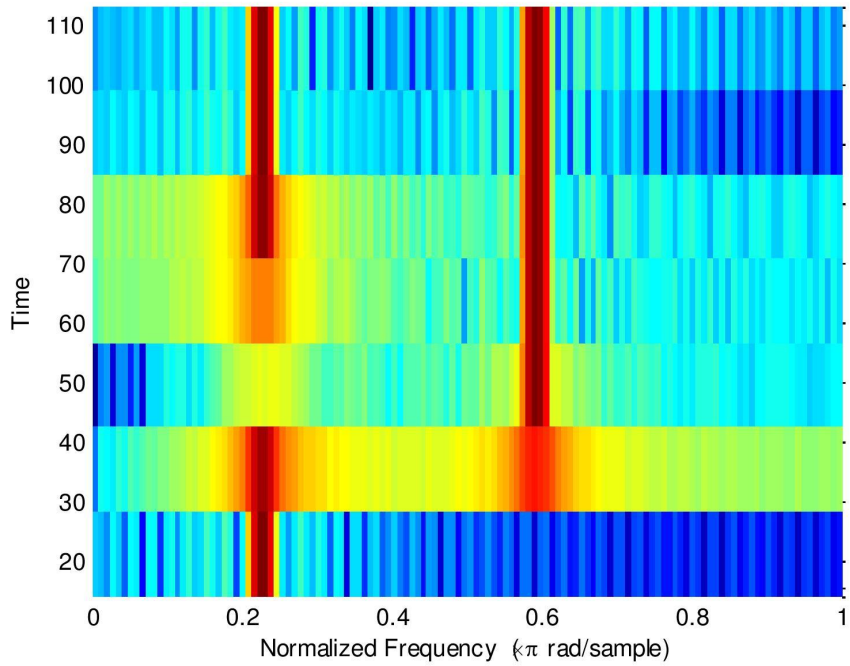




g) The filter has a sharp decaying characteristic above w=0.25 π. Hence it filters out the cosine component in the input signal that is situated at w=0.59 π, while the sine component at w=0.22 π is passed to the output.

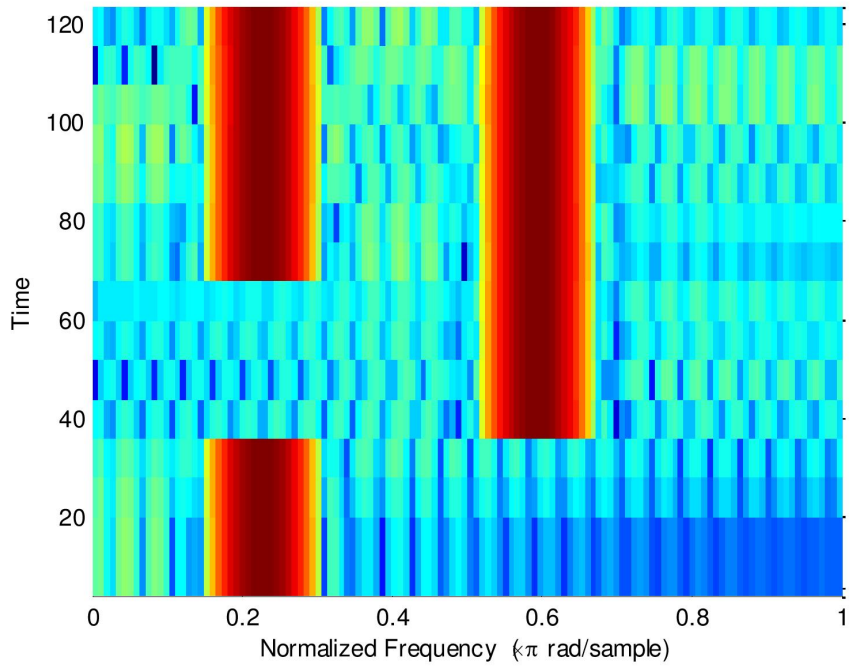
29)

a, b)



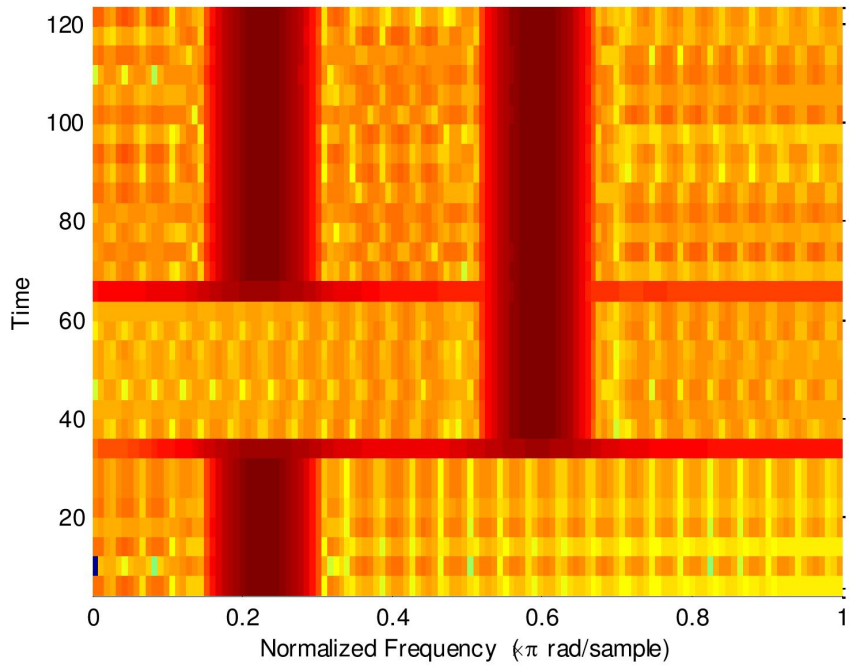



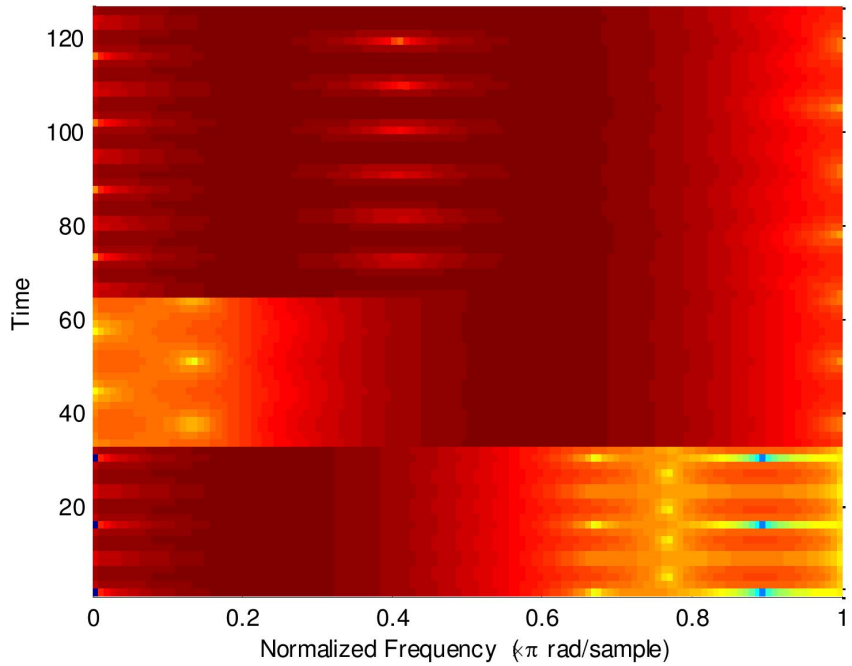

We observe the changes in the signal x both in time and frequency domains through the spectrogram function's output. In this way, different from directly observing the magnitude of the Fourier transform (abs(fft(x)) of x, we have the opportunity to investigate when a specific spectral component appears and when it ends in a signal. On the other hand with simple fft command we can only tell that a spectral component exists in that signal. The color coding in the spectrogram in MATLAB by default is as follow: the pixels close to red denote high power at that time-frequency pair and the pixels with color close to blue (purple) correspond to very low power on that grid. As given in the figure above, the low-frequency component with w=0.22π is present in the signal from normalized time instance 0 to 0.25 and from 0.50 to 1. It is absent in the duration (0.25, 0.50) times the length of simulation duration. Similarly, the high-frequency component at w=0.58π is identified to be present in the signal in the duration (0.25, 1) via the spectrogram analysis.




With "`spectrogram(x,50,0)`" command we obtain the above spectrogram, where we utilize a window of length 50 on the input signal and we make use of no overlapping between consecutive windowed sections of it. When we use spectrogram(x) without further defining the window length and overlapping, default window length (over which fft will be obtained for all time segments of x) is taken as 256 and consecutive windows are overlapped by half (50 % overlapping parameter). Here, by shortening the window length to 50 we increase number time windows and improve time resolution (where exactly a component starts and ends in time), however we worsen the frequency resolution (in frequency, wider components are observed in the spectrogram). This is a trade-off due to the well-known uncertainty principle, which dictates that we cannot know the time and the frequency properties of a signal component arbitrarily well under the same analysis.

When we run the "`spectrogram(x,50,25)`" command, 25% overlapping between time windows of length 50 is forced in the analysis. The use of overlapped windows results in smoother transitions in time, however also results in contamination of frequency bands close to the spectral components.




The command "`spectrogram(x,10,0)`" takes 10-point FFT's over 80 non-overlapping windows to identify the power content of the input signal at various time-frequency pairs. This way the resolution in time is high (we know where a spectral component starts and ends in time); however, we almost completely lost the capability for identification of frequency of the related components.