Digital Signal Processing Laboratory

Experiment 2

By: Hardik Tibrewal (18EC10020)

Designing Low Pass Filters By Windowing Method

Objectives:

(a) To design FIR (finite impulse response) filters for various orders and cut-off frequencies
(b) To test if pass-band and stop-band frequencies are attenuated in the filter designed
(c) To test how the designed filters respond to input signals contaminated by noise

Theoretical Background:

Ideally, one would prefer filters with a sharp cut-off since the accuracy is much higher and the control over pass-band and stop-band frequencies is highly desirable in digital signal processing. However, they are not practically realisable since their impulse responses will extend to infinity. So, in order to realise practical filters, we need to truncate the impulse response in time domain, which leads to a less-than-ideal response in the frequency domain. However, this compromise is necessary, and more importantly, it is an acceptable one.

We can apply different kinds of windows in the time domain, and they will show different responses in the frequency domain, leading to better or worse filter characteristics, depending upon the type of window. The windows we have simulated in this experiment are the following: Rectangular, Triangular, Hanning, Hamming, and Blackmann.

If the desired filter frequency response is $H_d(\omega)$ and its corresponding impulse response is $h_d(n)$, then by the equations of Discrete-Time Fourier Transform:

$$H_d(\omega) = \sum_{n=-\infty}^{\infty} h_d(n) e^{-j\omega n}$$

$$\mathrm{h_d(n)} = \frac{1}{2\pi}\int_{-\pi}^{\pi} \mathrm{H_d(\omega) e^{j\omega n} d\omega}$$

So, if the window's frequency response is W(ω) and the time domain sequence is w(n) (non-zero only between 0 and N-1), then

$$W(\omega) = \sum_{n=0}^{\mathrm{N-1}} w(n) e^{-j\omega n}$$

$$w(n) = \frac{1}{2\pi}\int_{-\pi}^{\pi} W(\omega) e^{j\omega n} d\omega$$

Using these, our actual filter response H(ω), and its impulse response h(n), will follow the following equations:

$$h(n) = w(n) \cdot h_d(n)$$

$$\mathrm{H(\omega)} = \frac{1}{2\pi}\int_{-\pi}^{\pi} H_d(\nu) W(\omega - \nu) d\nu$$

The time domain window functions are as follows:

$$\begin{aligned}
\text{Rectangular Window} : w(n) &= 1, \ \mathrm{n} = 0,1,\ldots,\mathrm{N\text{-}1} \\
&= 0, \ \text{otherwise} \\
\text{Triangular Window} : w(n) &= 1 - 2 \cdot \left(\frac{|n - (N-1)/2|}{N-1} \right), \ \mathrm{n} = 0,1,\ldots,\mathrm{N\text{-}1} \\
&= 0, \ \text{otherwise} \\
\text{Hanning Window} : w(n) &= 0.5 - 0.5 \cdot cos\left(\frac{2\pi n}{N-1} \right), \ \mathrm{n} = 0,1,\ldots,\mathrm{N\text{-}1} \\
&= 0, \ \text{otherwise} \\
\text{Hamming Window} : w(n) &= 0.54 - 0.46 \cdot cos\left(\frac{2\pi n}{N-1} \right), \ \mathrm{n} = 0,1,\ldots,\mathrm{N\text{-}1} \\
&= 0, \ \text{otherwise} \\
\text{Blackmann Window} : w(n) &= 0.42 - 0.5 \cdot cos\left(\frac{2\pi n}{N-1} \right) \\
&\quad + 0.08 \cdot cos\left(\frac{4\pi n}{N-1} \right), \ \mathrm{n} = 0,1,\ldots,\mathrm{N\text{-}1} \\
&= 0, \ \text{otherwise}
\end{aligned}$$

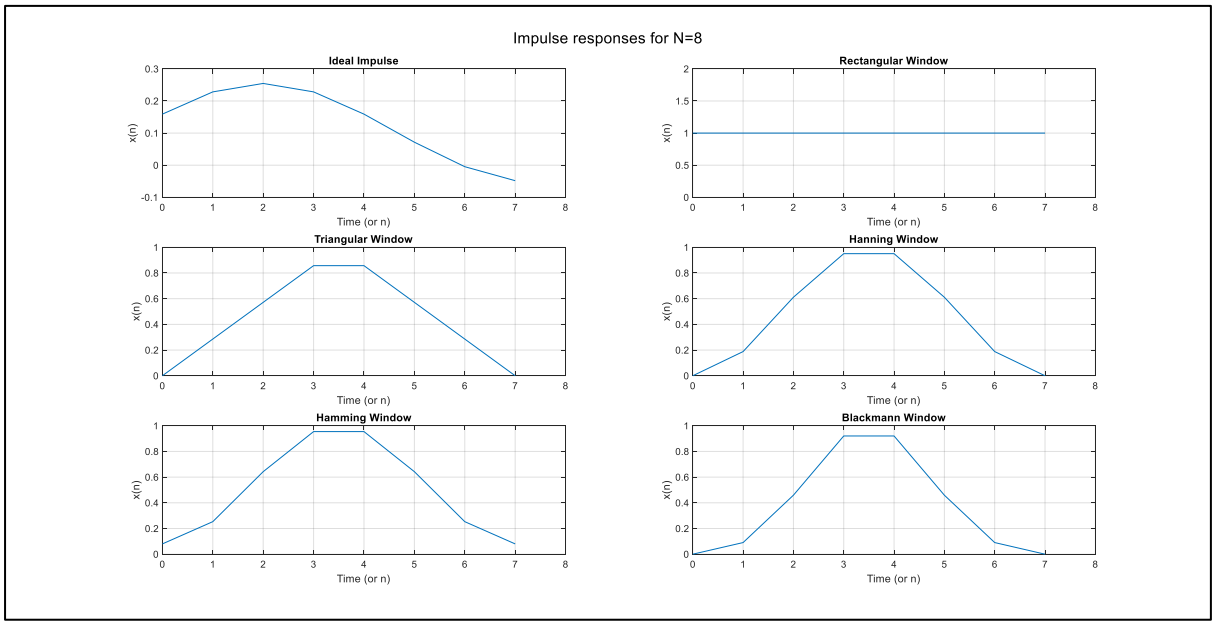
Pseudocode:

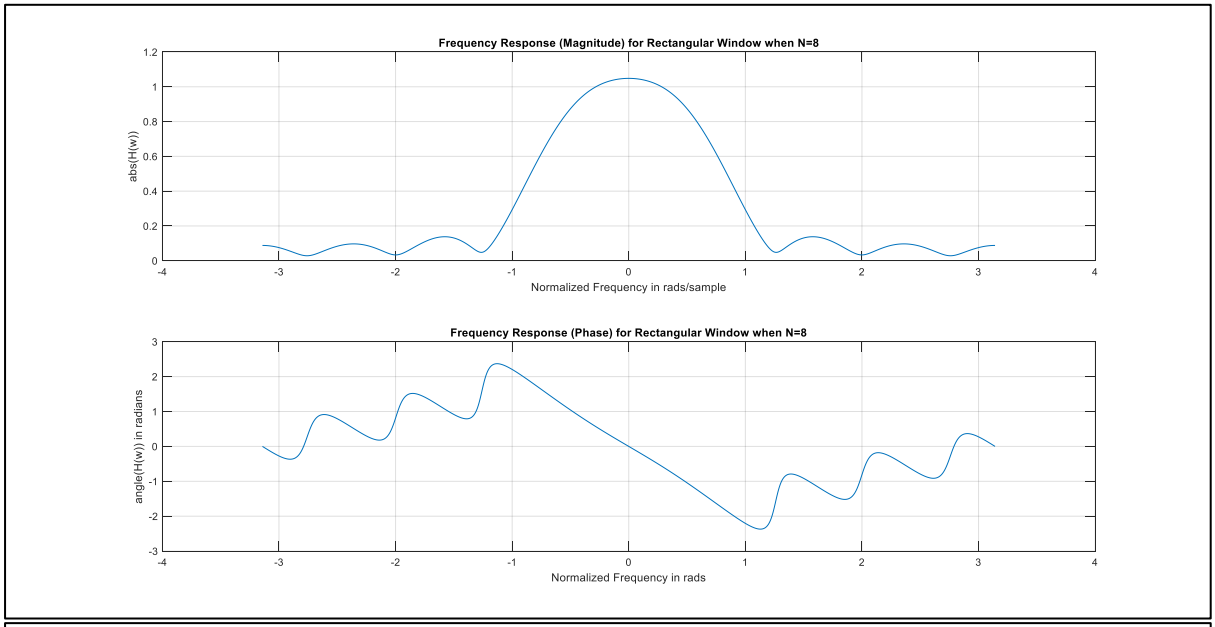
For generating the windowed filter functions in time domain, we first defined the ideal filter's impulse function and the window functions. The ideal impulse actually extends to infinite time, but since any part outside the window will become zero, we do not need to define additional points. Then the signals were multiplied element-wise, and then their frequency domain responses were plotted.

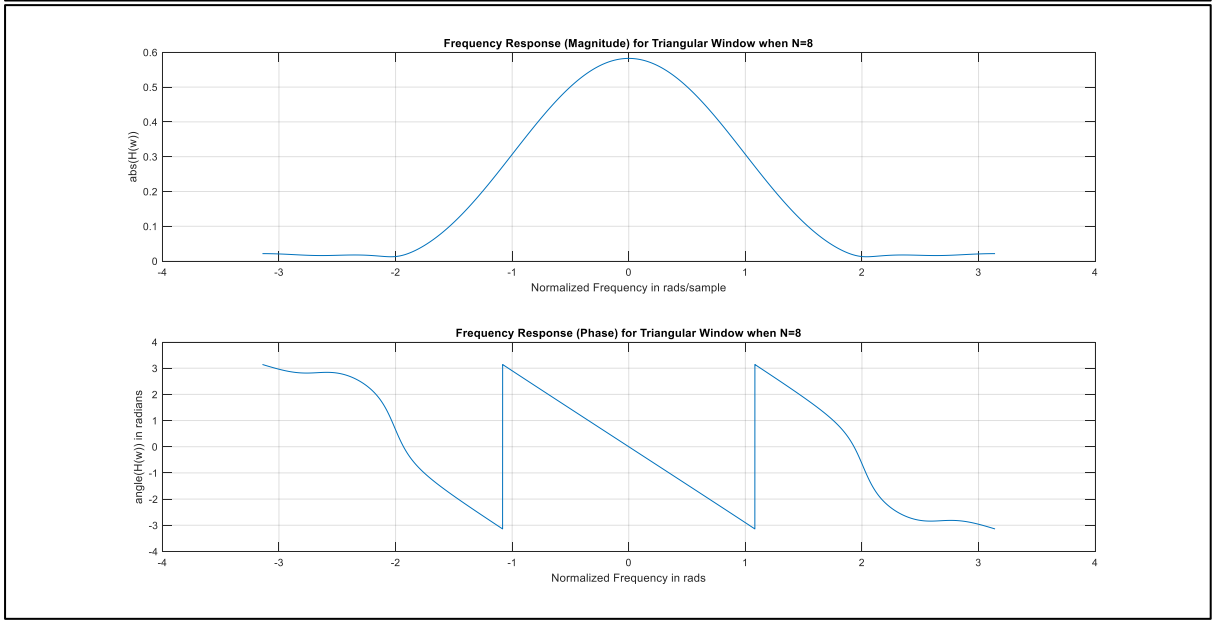
Then a simple signal was generated which had two frequency components: one in the pass-band and the other in the stop-band. Some random noise was added to the signal such that the input SNR was 20 dB. Then the signal was passed through the various filters using the *filtfilt* function and the output was observed. Along with this, the output SNR was also calculated.

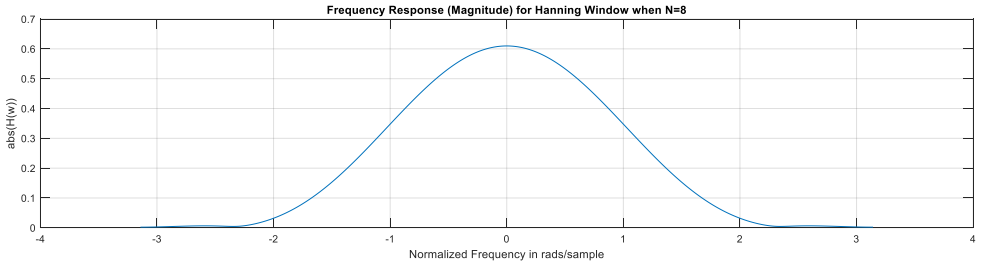
Simulation Results:

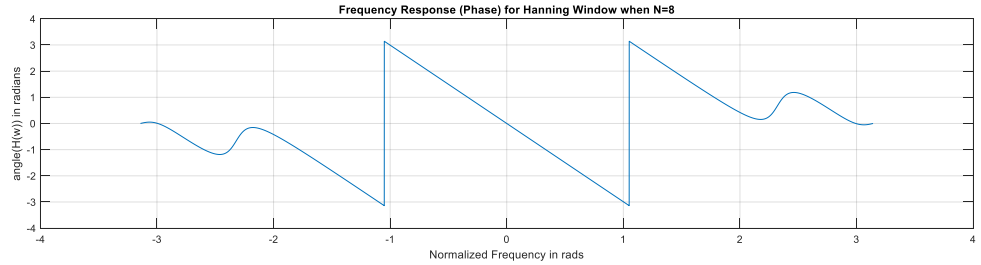
(From next page)

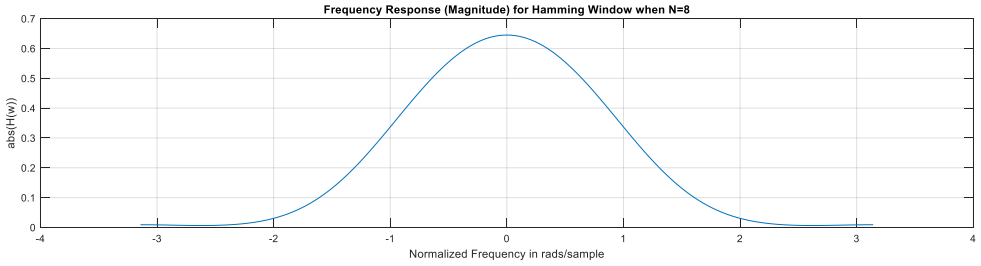

Impulse responses for N=8
Ideal Impulse
Rectangular Window
Triangular Window
Hanning Window
Hamming Window
Blackmann Window
x(n)
Time (or n)

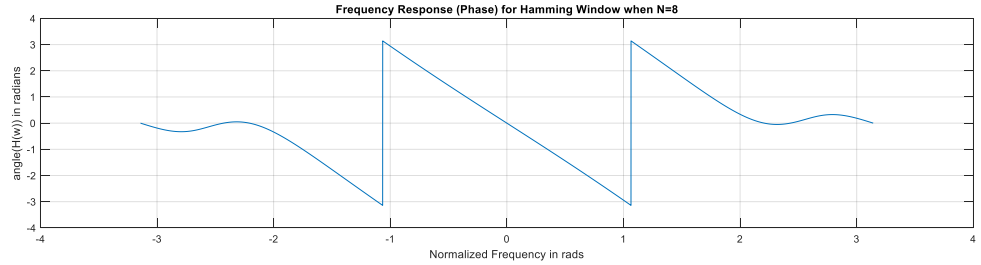

Frequency Response (Magnitude) for Rectangular Window when N=8
abs(H(w))
Normalized Frequency in rads/sample
Frequency Response (Phase) for Rectangular Window when N=8
angle(H(w)) in radians
Normalized Frequency in rads

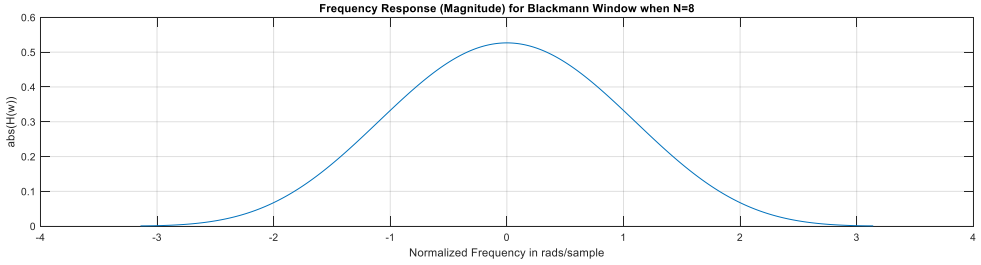

Frequency Response (Magnitude) for Triangular Window when N=8
abs(H(w))
Normalized Frequency in rads/sample
Frequency Response (Phase) for Triangular Window when N=8
angle(H(w)) in radians
Normalized Frequency in rads

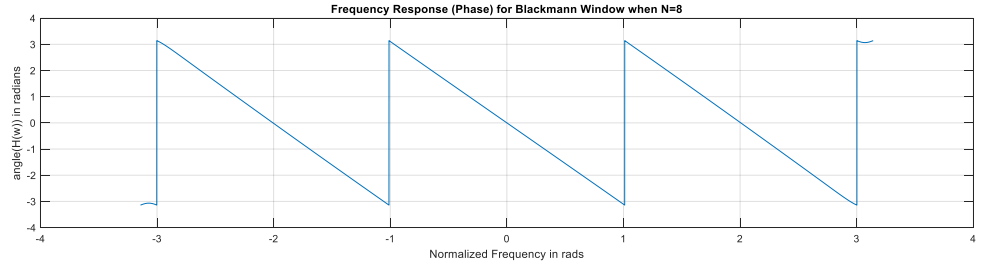

Frequency Response (Magnitude) for Hanning Window when N=8
abs(H(w))
Normalized Frequency in rads/sample

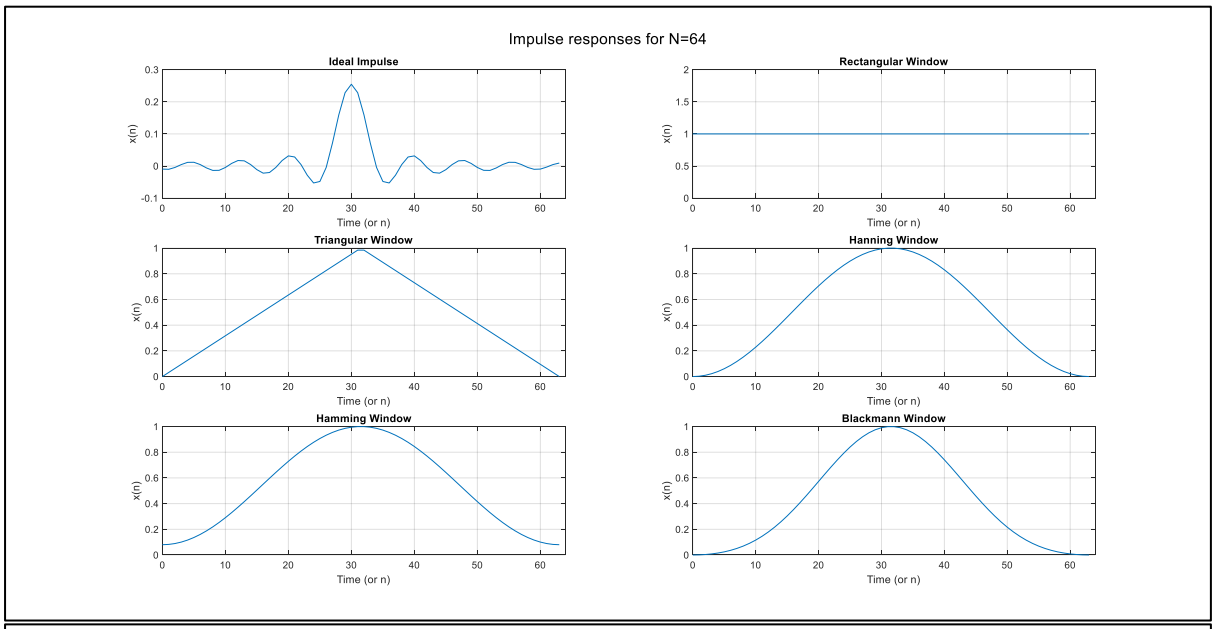

Frequency Response (Phase) for Hanning Window when N=8
angle(H(w)) in radians
Normalized Frequency in rads

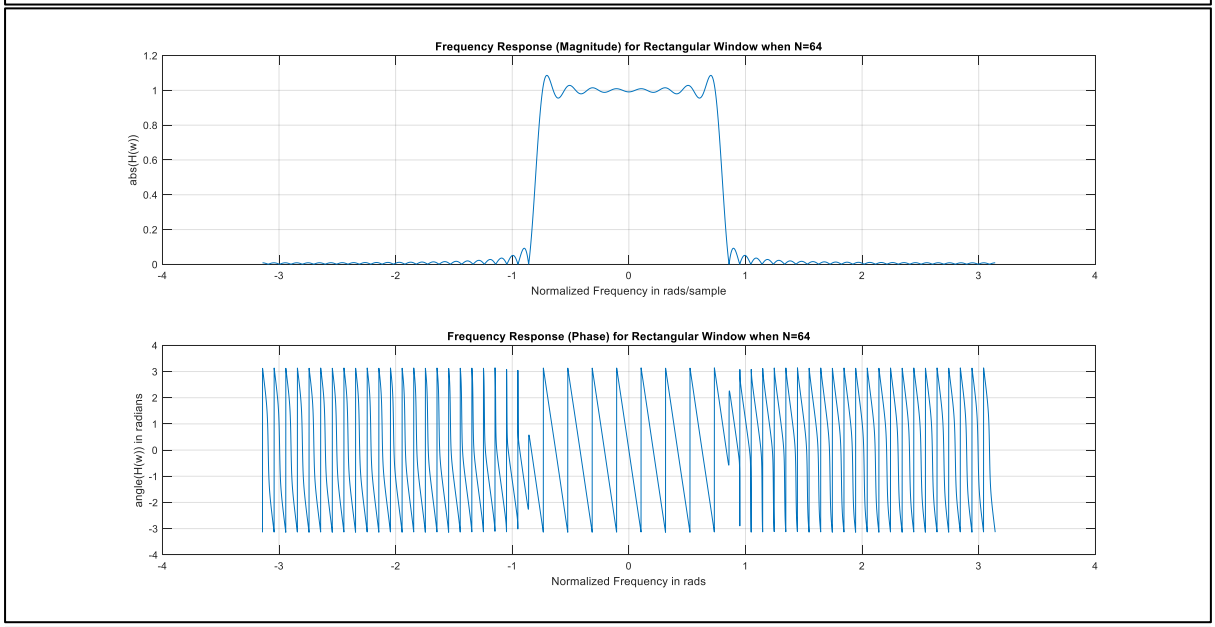

Frequency Response (Magnitude) for Hamming Window when N=8
abs(H(w))
Normalized Frequency in rads/sample

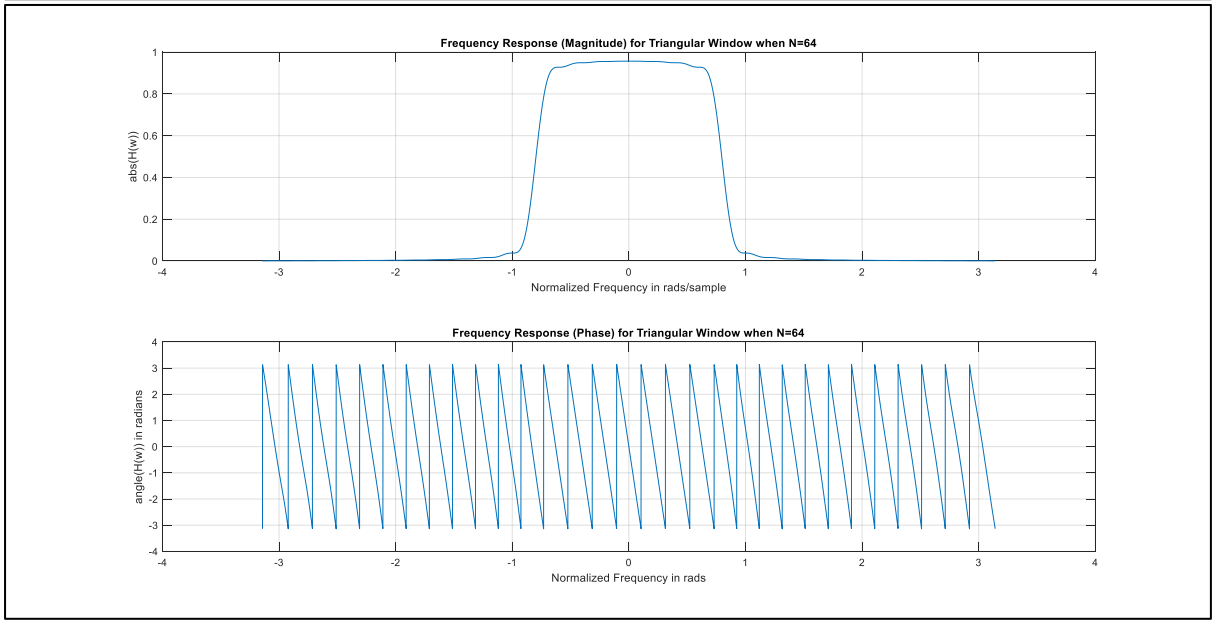

Frequency Response (Phase) for Hamming Window when N=8
angle(H(w)) in radians
Normalized Frequency in rads

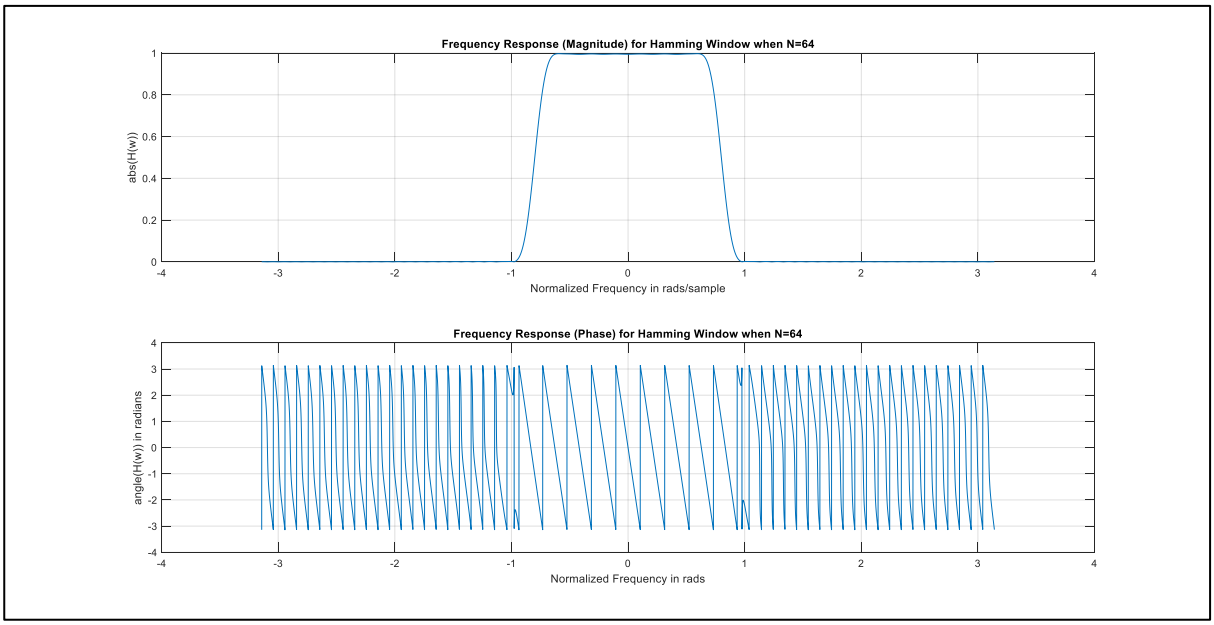

Frequency Response (Magnitude) for Blackmann Window when N=8
abs(H(w))
Normalized Frequency in rads/sample

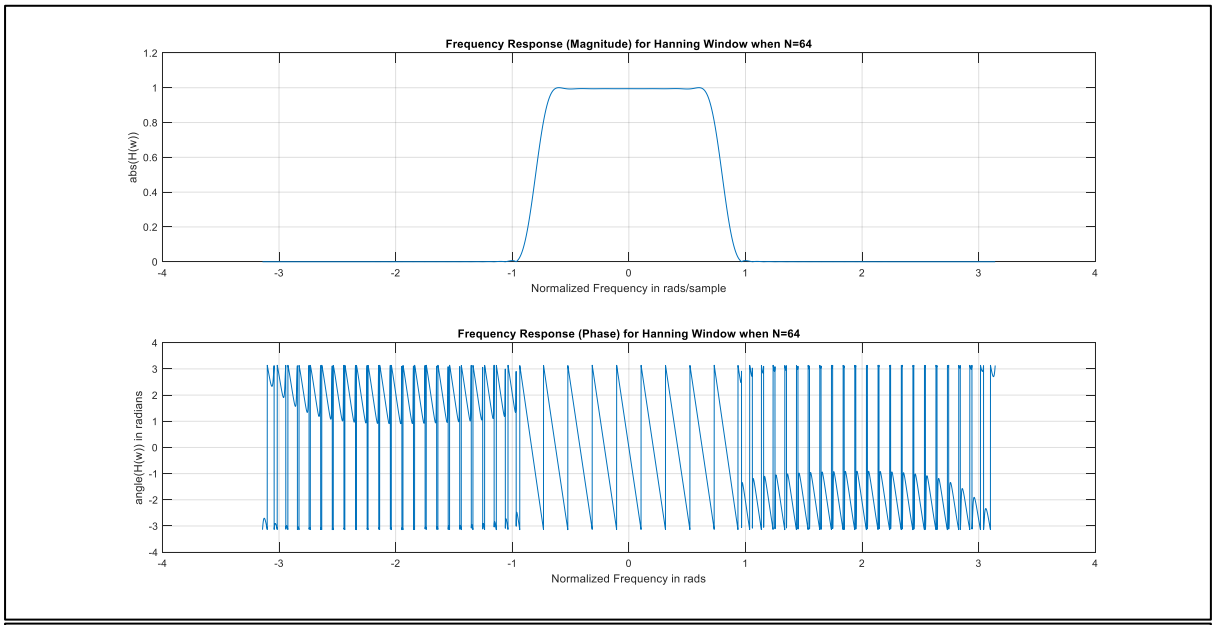

Frequency Response (Phase) for Blackmann Window when N=8
angle(H(w)) in radians
Normalized Frequency in rads

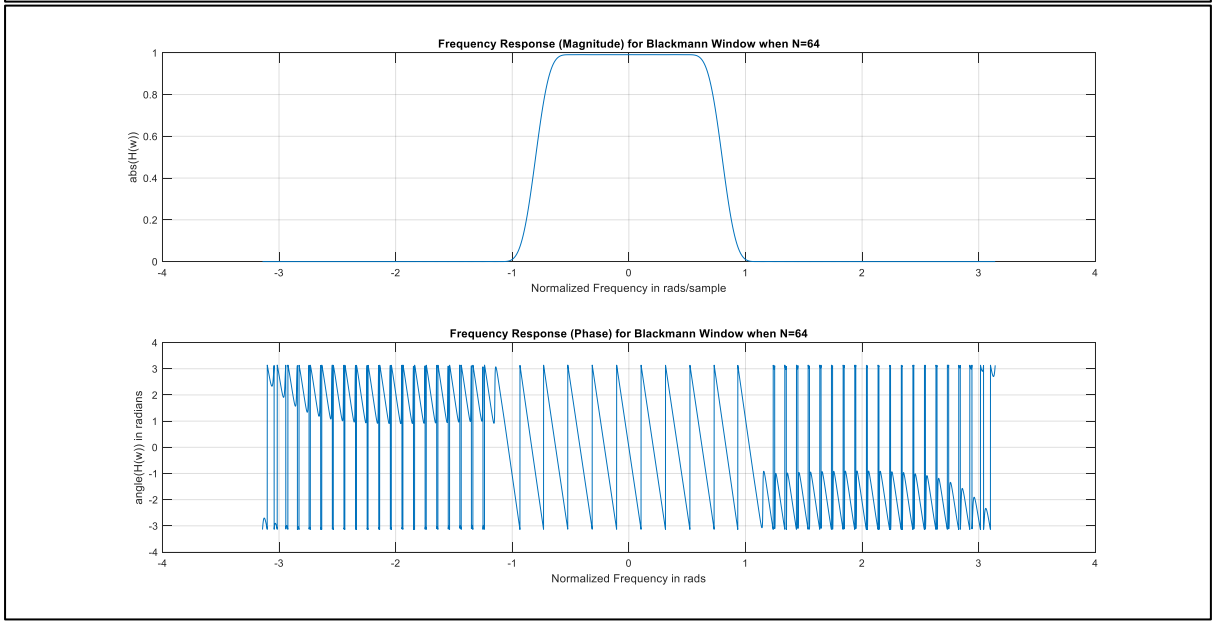

Impulse responses for N=64
Ideal Impulse
Rectangular Window
Triangular Window
Hanning Window
Hamming Window
Blackmann Window
x(n)
Time (or n)

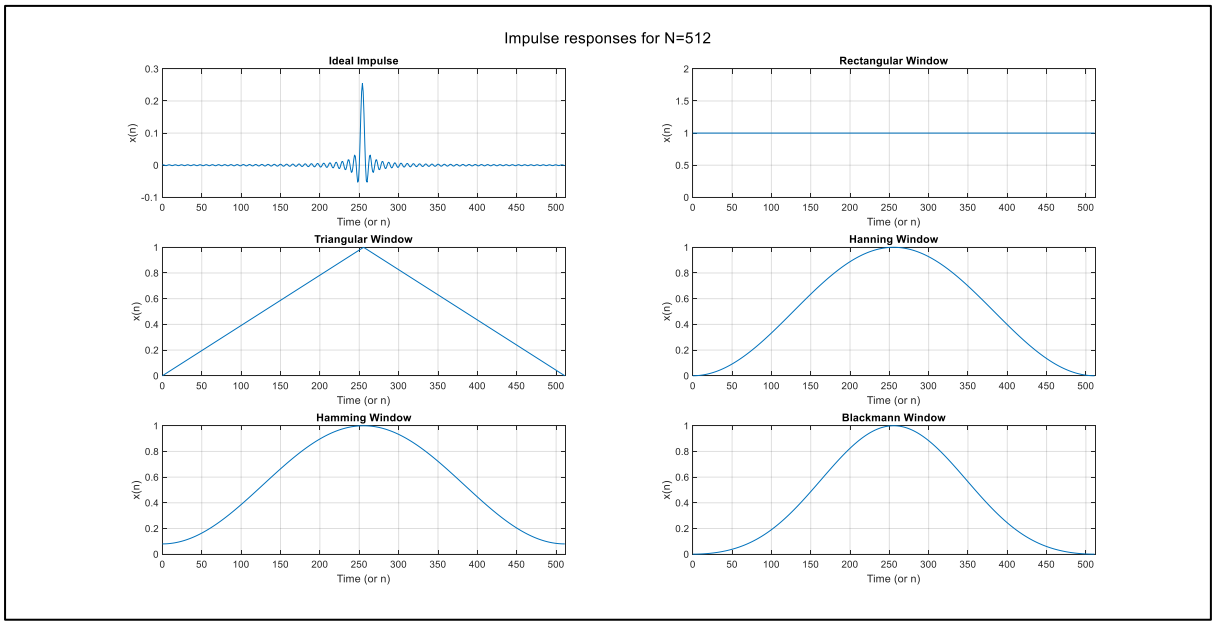

Frequency Response (Magnitude) for Rectangular Window when N=64
abs(H(w))
Normalized Frequency in rads/sample
Frequency Response (Phase) for Rectangular Window when N=64
angle(H(w)) in radians
Normalized Frequency in rads

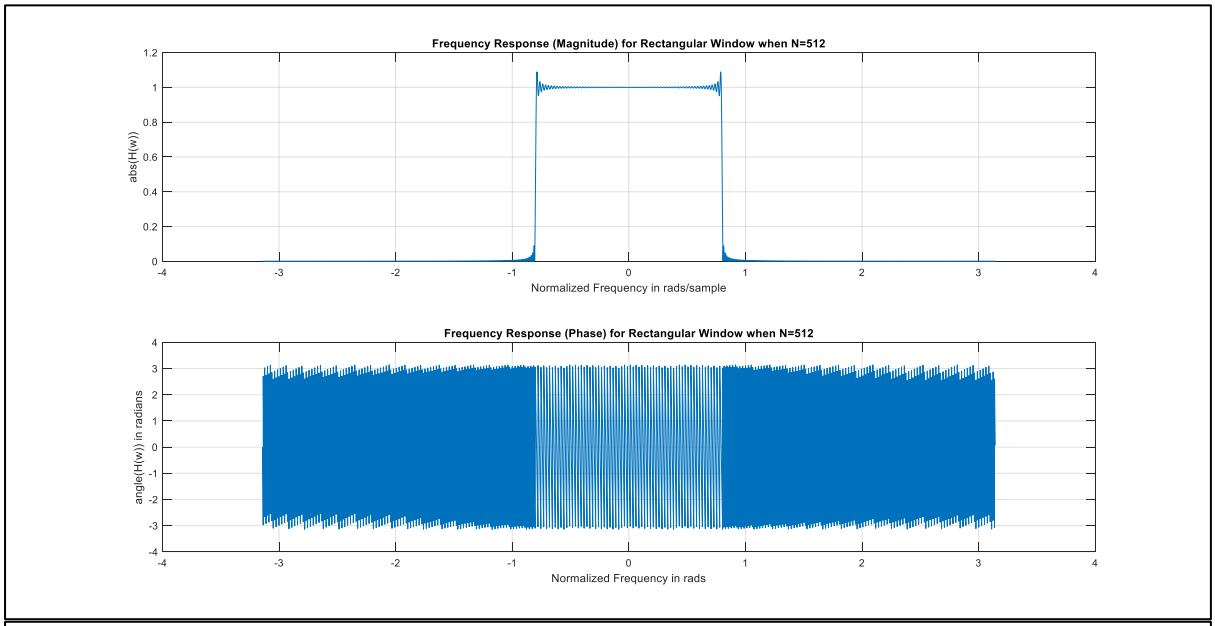

Frequency Response (Magnitude) for Triangular Window when N=64
abs(H(w))
Normalized Frequency in rads/sample
Frequency Response (Phase) for Triangular Window when N=64
angle(H(w)) in radians
Normalized Frequency in rads

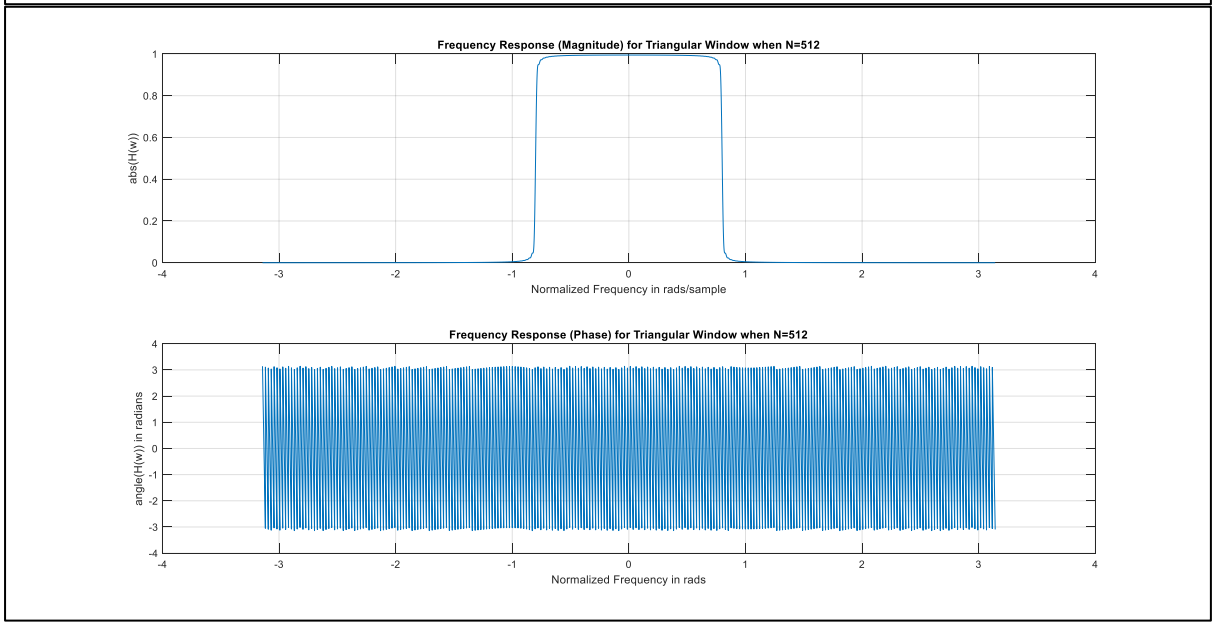

Frequency Response (Magnitude) for Hamming Window when N=64
abs(H(w))
Normalized Frequency in rads/sample
Frequency Response (Phase) for Hamming Window when N=64
angle(H(w)) in radians
Normalized Frequency in rads

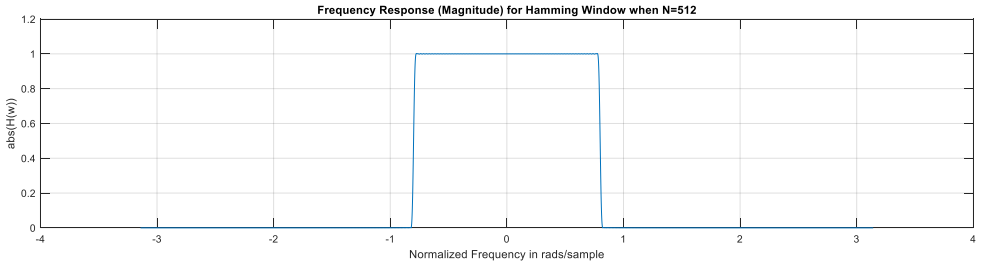

Frequency Response (Magnitude) for Hanning Window when N=64
abs(H(w))
Normalized Frequency in rads/sample
Frequency Response (Phase) for Hanning Window when N=64
angle(H(w)) in radians
Normalized Frequency in rads

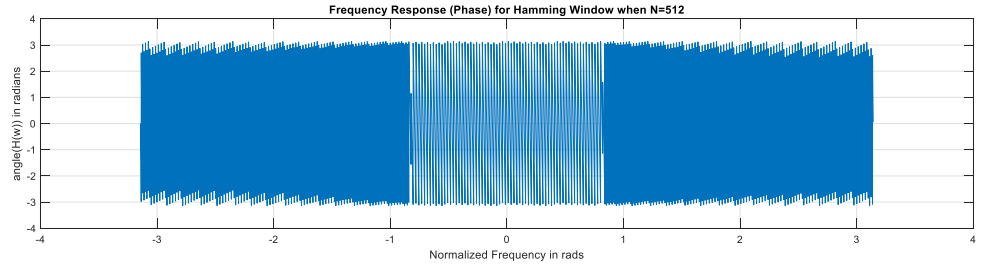

Frequency Response (Magnitude) for Blackmann Window when N=64
abs(H(w))
Normalized Frequency in rads/sample
Frequency Response (Phase) for Blackmann Window when N=64
angle(H(w)) in radians
Normalized Frequency in rads

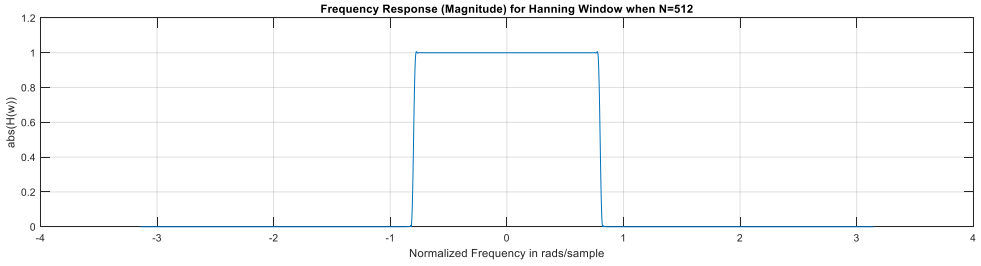

Impulse responses for N=512
Ideal Impulse
Rectangular Window
Triangular Window
Hanning Window
Hamming Window
Blackmann Window
x(n)
Time (or n)

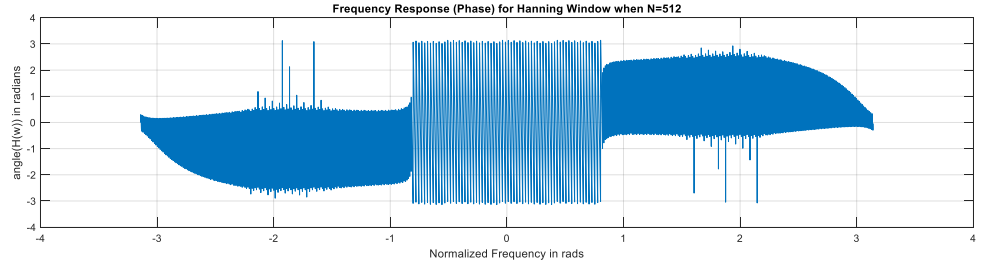

Frequency Response (Magnitude) for Rectangular Window when N=512
abs(H(w))
Normalized Frequency in rads/sample
Frequency Response (Phase) for Rectangular Window when N=512
angle(H(w)) in radians
Normalized Frequency in rads

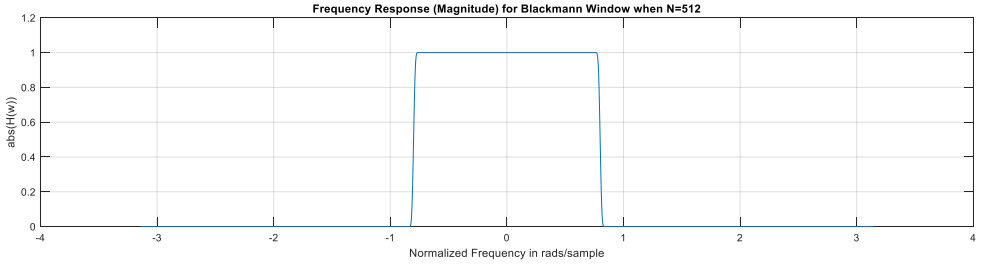

Frequency Response (Magnitude) for Triangular Window when N=512
abs(H(w))
Normalized Frequency in rads/sample
Frequency Response (Phase) for Triangular Window when N=512
angle(H(w)) in radians
Normalized Frequency in rads

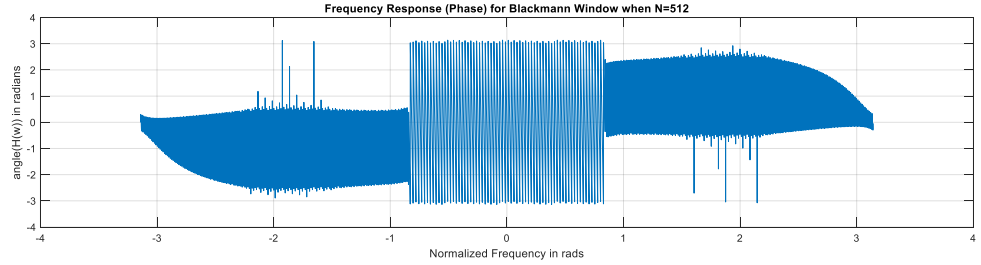

Frequency Response (Magnitude) for Hamming Window when N=512
abs(H(w))
Normalized Frequency in rads/sample


Frequency Response (Phase) for Hamming Window when N=512
angle(H(w)) in radians
Normalized Frequency in rads

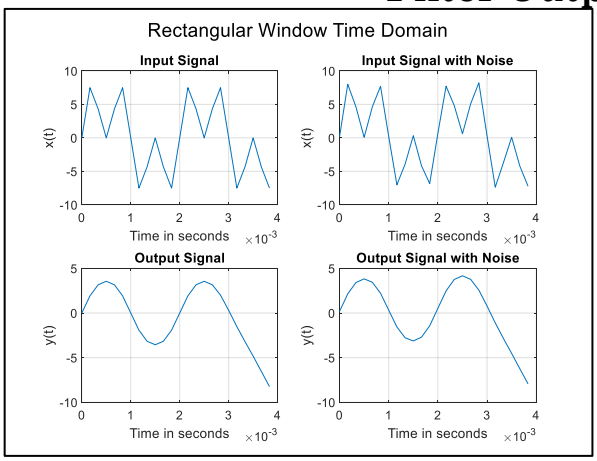

Frequency Response (Magnitude) for Hanning Window when N=512
abs(H(w))
Normalized Frequency in rads/sample

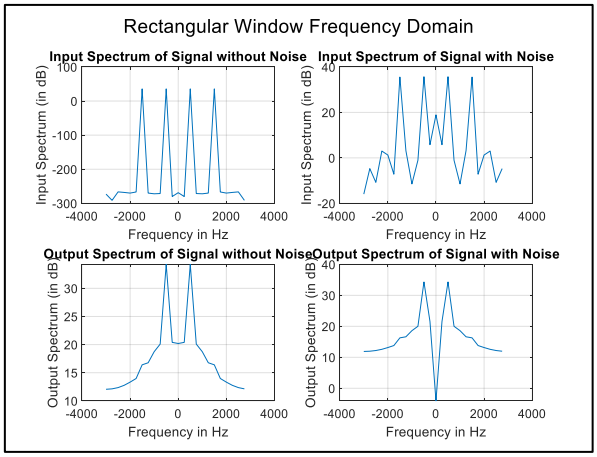

Frequency Response (Phase) for Hanning Window when N=512
angle(H(w)) in radians
Normalized Frequency in rads

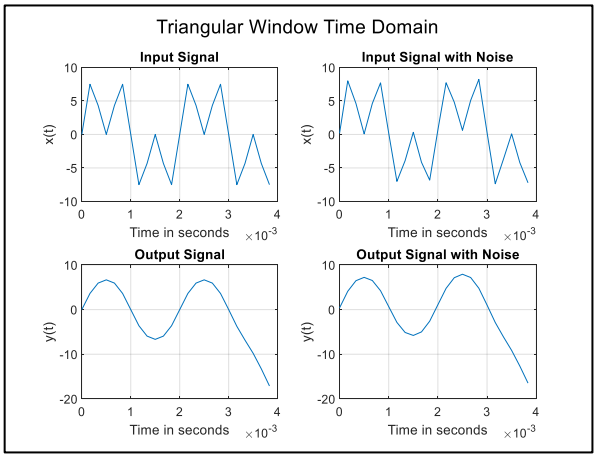

Frequency Response (Magnitude) for Blackmann Window when N=512
abs(H(w))
Normalized Frequency in rads/sample

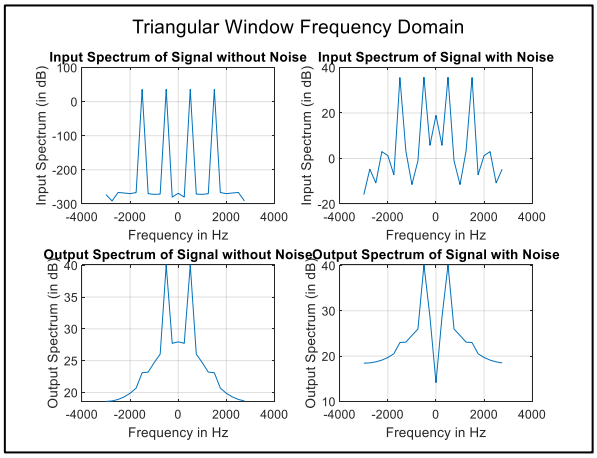

Frequency Response (Phase) for Blackmann Window when N=512
angle(H(w)) in radians
Normalized Frequency in rads

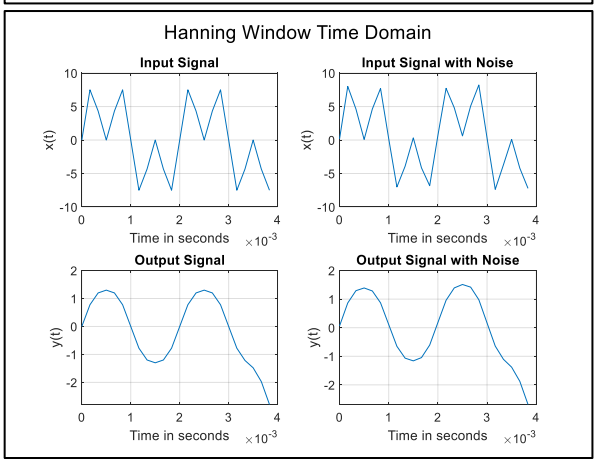
Filter Outputs for N=8

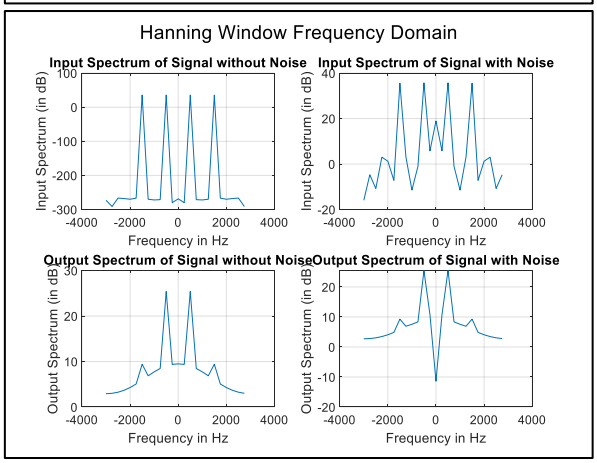


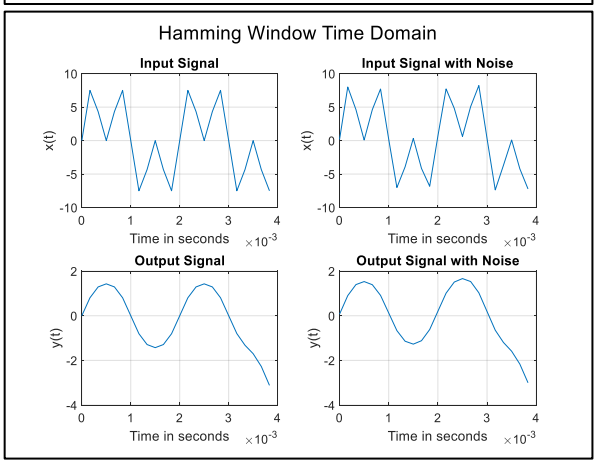


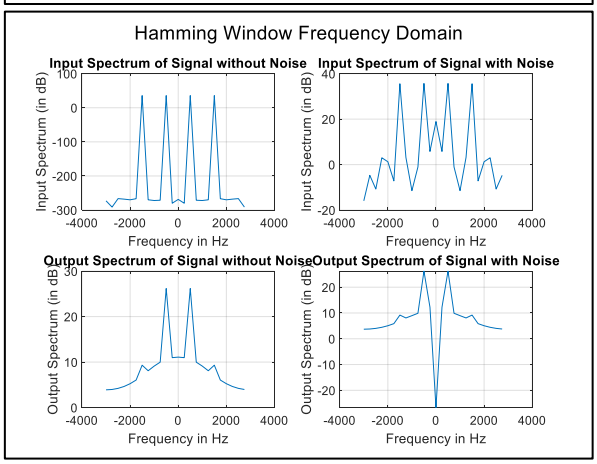


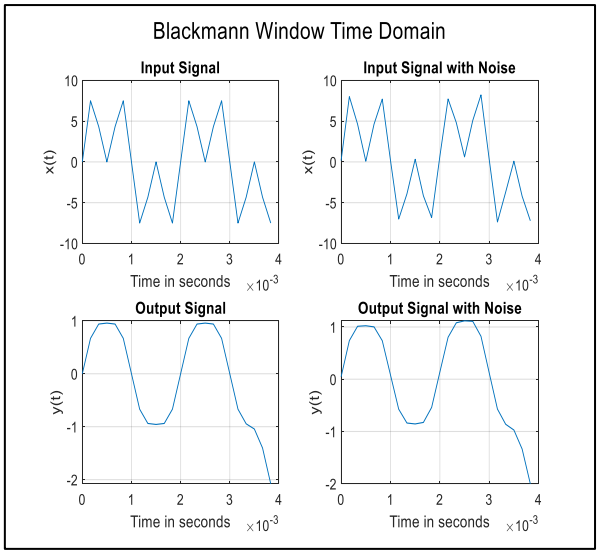


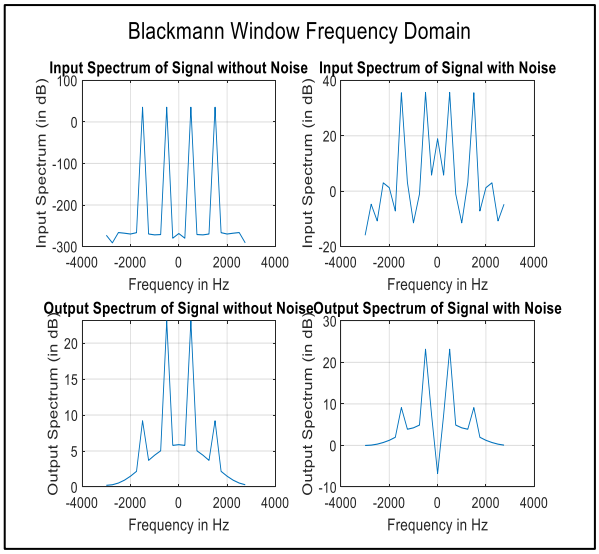




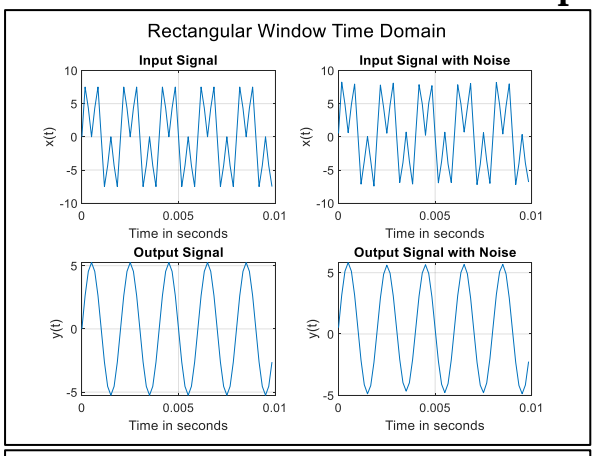


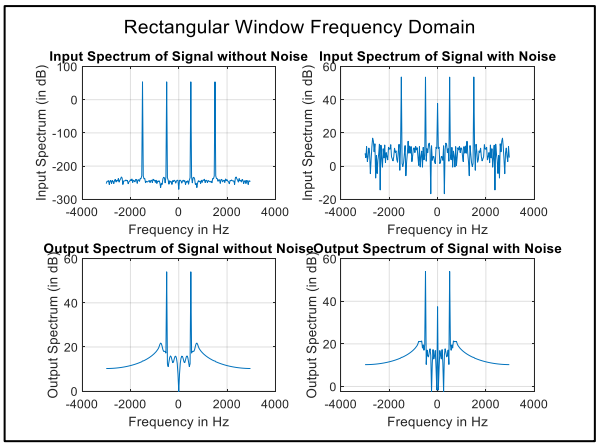


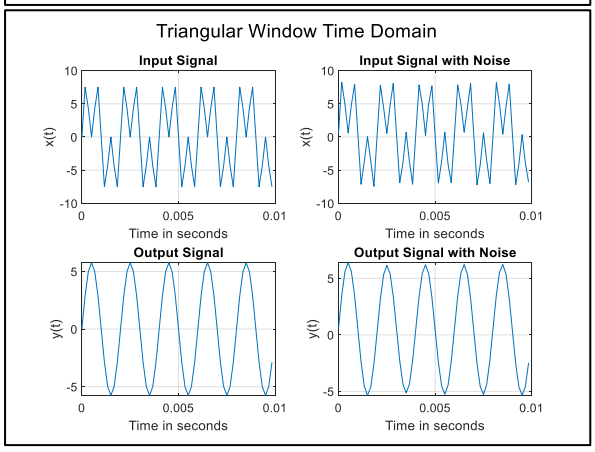

Blackmann Window Time Domain
Input Signal
Input Signal with Noise
Output Signal
Output Signal with Noise
x(t)
y(t)
Time in seconds
×10^-3

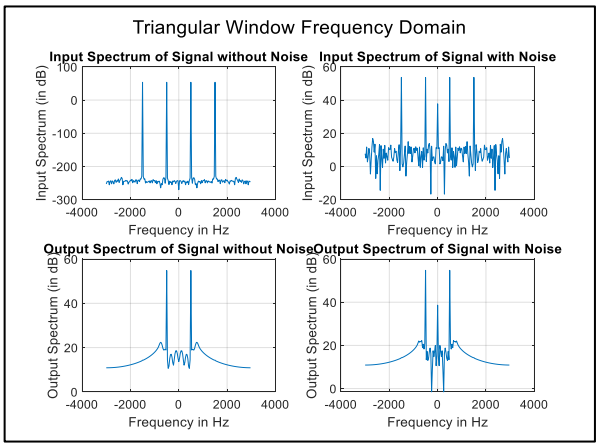

Blackmann Window Frequency Domain
Input Spectrum of Signal without Noise
Input Spectrum of Signal with Noise
Output Spectrum of Signal without Noise
Output Spectrum of Signal with Noise
Input Spectrum (in dB)
Output Spectrum (in dB)
Frequency in Hz

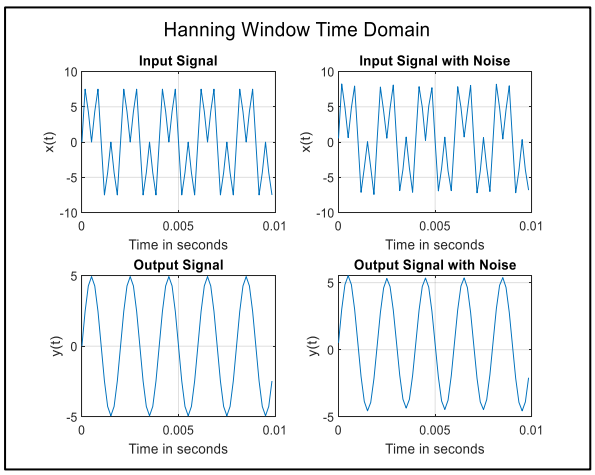
Filter outputs for N = 64

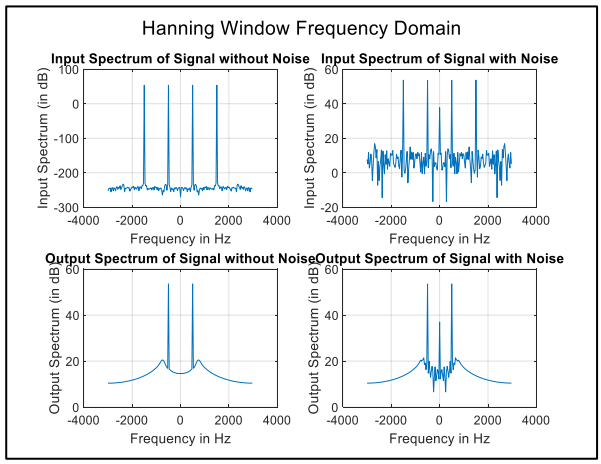


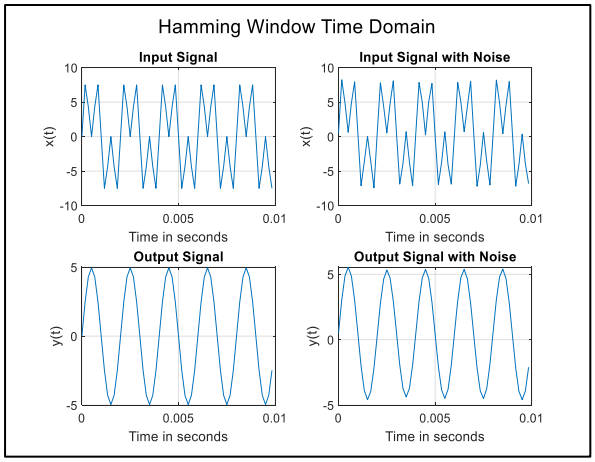


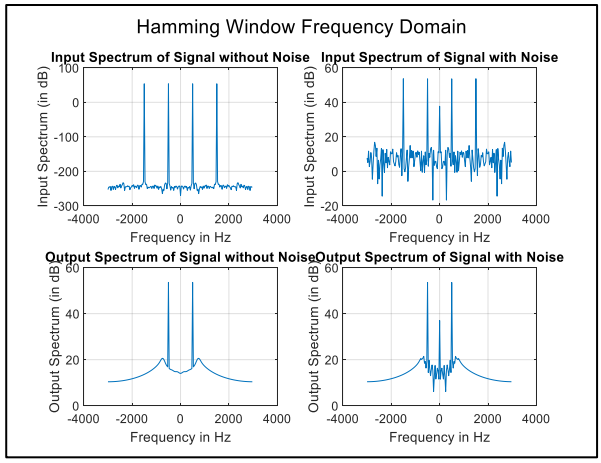


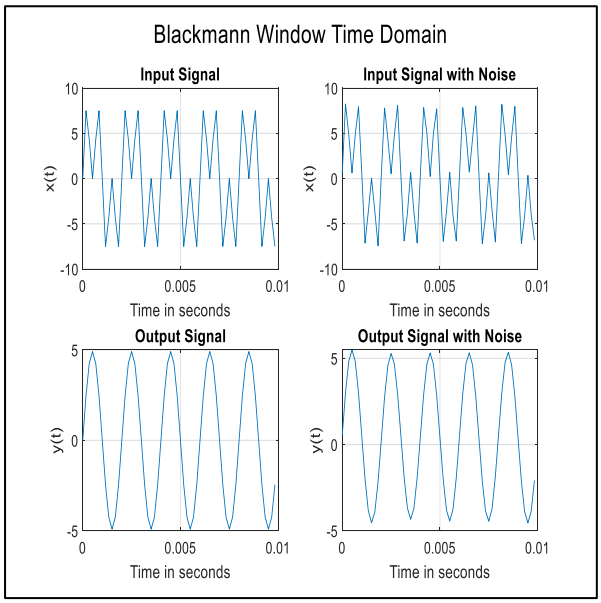


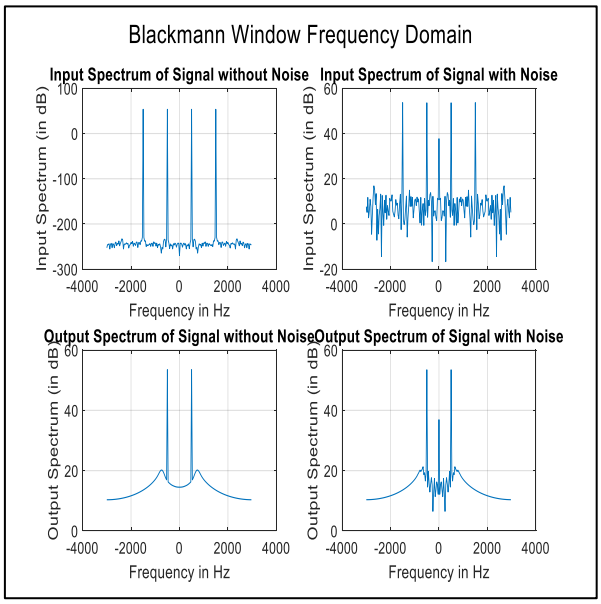




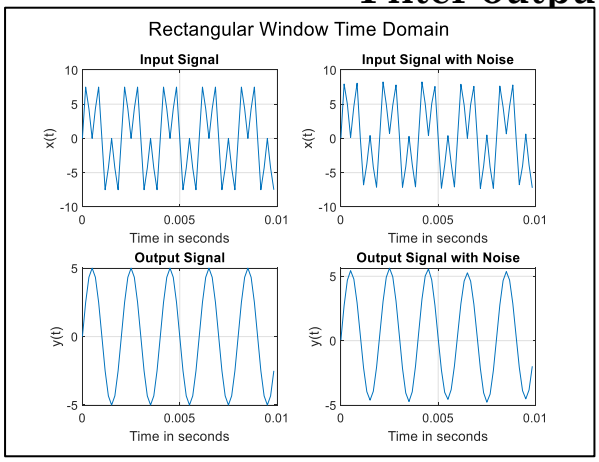


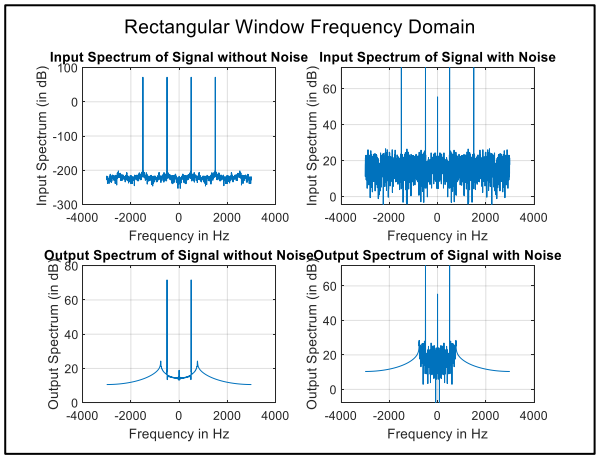


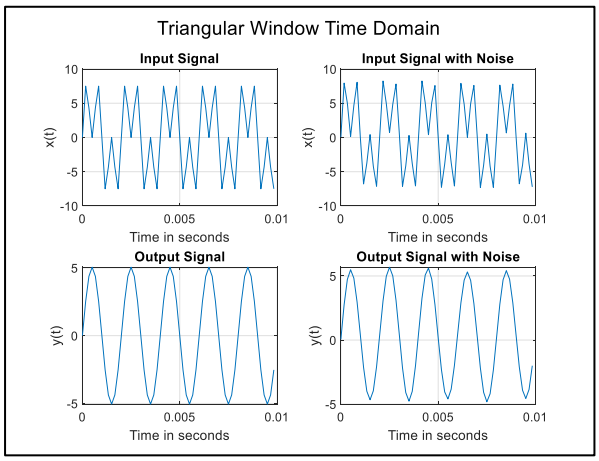

Blackmann Window Time Domain
Input Signal
Input Signal with Noise
Output Signal
Output Signal with Noise
x(t)
y(t)
Time in seconds

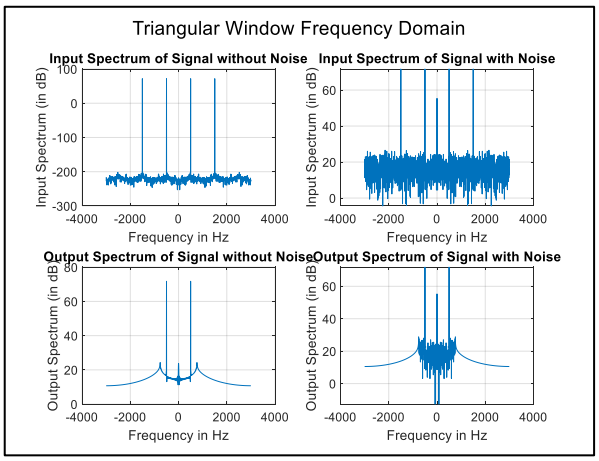

Blackmann Window Frequency Domain
Input Spectrum of Signal without Noise
Input Spectrum of Signal with Noise
Output Spectrum of Signal without Noise
Output Spectrum of Signal with Noise
Input Spectrum (in dB)
Output Spectrum (in dB)
Frequency in Hz

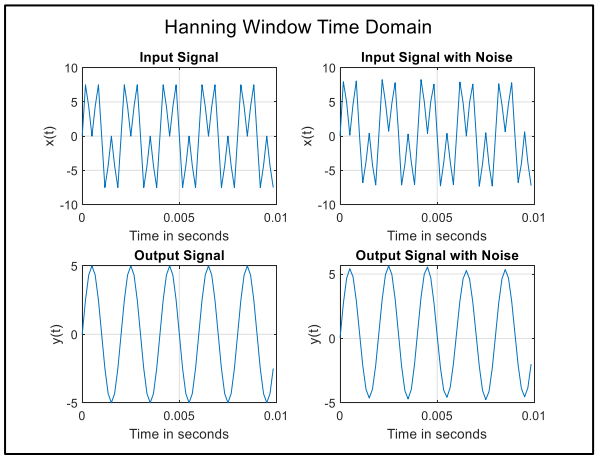
Filter outputs for N = 512

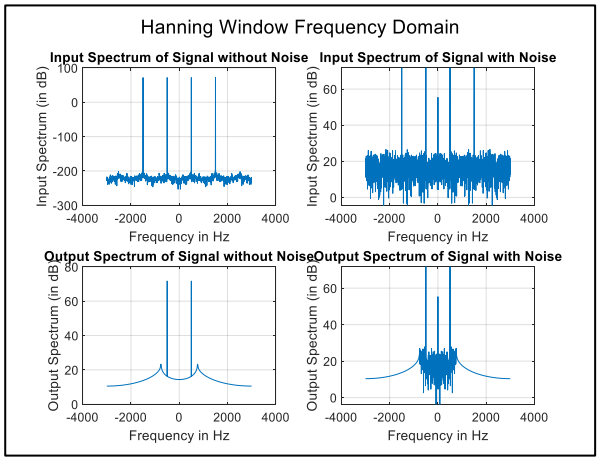


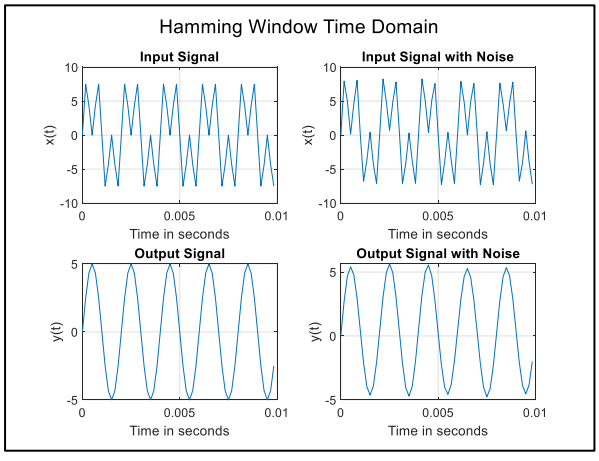


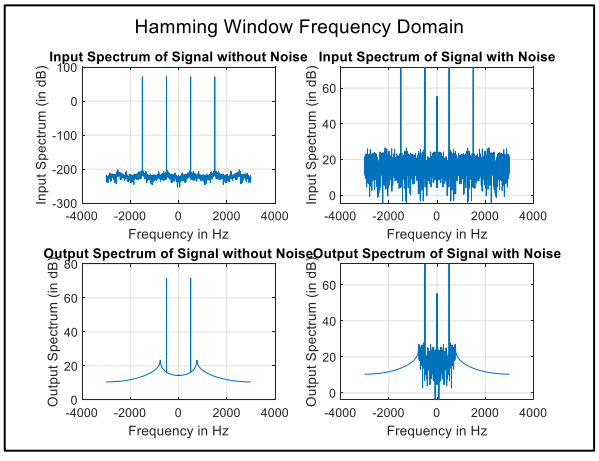


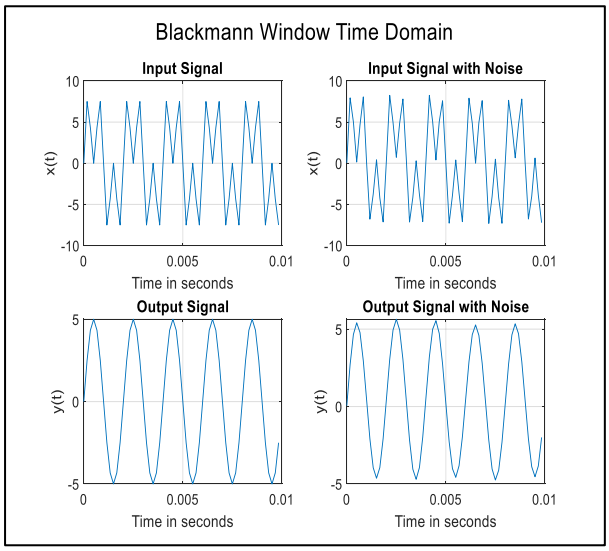


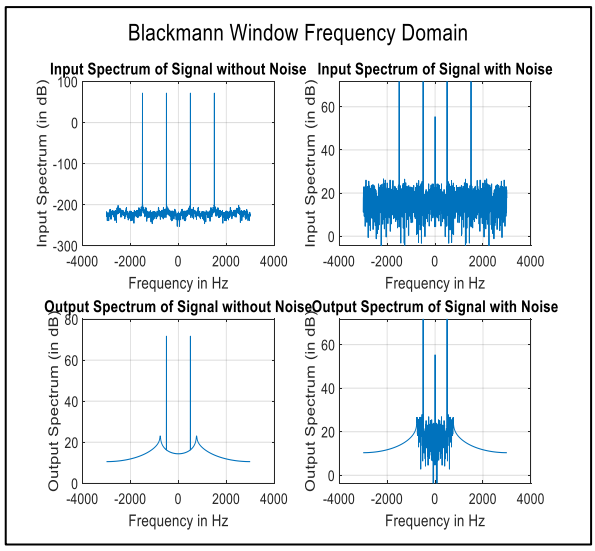









Blackmann Window Time Domain
Input Signal
Input Signal with Noise
Output Signal
Output Signal with Noise
x(t)
y(t)
Time in seconds


Blackmann Window Frequency Domain
Input Spectrum of Signal without Noise
Input Spectrum of Signal with Noise
Output Spectrum of Signal without Noise
Output Spectrum of Signal with Noise
Input Spectrum (in dB)
Output Spectrum (in dB)
Frequency in Hz

Tables:

Rectangular Windowing Characteristics

N	Transition Width (in π rads/sample)	Peak of first lobe (in dB)	Maximum stop-band attenuation (in dB)
8	0.19	-17.21	-30.73
64	0.025	-20.96	-57.25
512	0.013	-33.54	-71.40

Triangular Windowing Characteristics

N	Transition Width (in π rads/sample)	Peak of first lobe (in dB)	Maximum stop-band attenuation (in dB)
8	0.41	-35.15	-38.13
64	0.266	-43.99	-67.27
512	N/A	N/A	-82.85

Hanning Windowing Characteristics

N	Transition Width (in π rads/sample)	Peak of first lobe (in dB)	Maximum stop-band attenuation (in dB)
8	0.49	-43.80	-53.30
64	0.066	-44.11	-124.83
512	0.01	-48.47	-184.10

Hamming Windowing Characteristics

N	Transition Width (in π rads/sample)	Peak of first lobe (in dB)	Maximum stop-band attenuation (in dB)
8	0.61	-40.89	-43.87
64	0.068	-54.43	-82.32
512	0.01	-58.24	-93.39

Blackmann Windowing Characteristics

N	Transition Width (in π rads/sample)	Peak of first lobe (in dB)	Maximum stop-band attenuation (in dB)
8	0.71	No side lobe	-61.04
64	0.127	-75.22	-133.07
512	0.014	-77.09	-192.97

Rectangular Window Outputs

N	Signal Amplitude A_S (in V)	Noise Amplitude A_N (in V)	SNR $20*\log_{10}(A_S/A_N)$ (in dB)	SNR using MATLAB Function (in dB)
8	3.55	0.71	14.03	16.98
64	5.45	0.70	17.86	19.69
512	5.16	0.69	17.50	19.11

Triangular Window Outputs

N	Signal Amplitude A_S (in V)	Noise Amplitude A_N (in V)	SNR $20*\log_{10}(A_S/A_N)$ (in dB)	SNR using MATLAB Function (in dB)
8	6.67	1.47	13.15	16.57
64	6.09	0.76	18.10	19.85
512	5.18	0.70	17.42	19.06

Hanning Window Outputs

N	Signal Amplitude A_S (in V)	Noise Amplitude A_N (in V)	SNR $20*\log_{10}(A_S/A_N)$ (in dB)	SNR using MATLAB Function (in dB)
8	1.30	0.24	14.71	17.20
64	5.04	0.69	17.23	19.18
512	5.19	0.69	17.51	19.12

Hamming Window Outputs

N	Signal Amplitude A_S (in V)	Noise Amplitude A_N (in V)	SNR $20*\log_{10}(A_S/A_N)$ (in dB)	SNR using MATLAB Function (in dB)
8	1.43	0.27	14.56	17.34
64	5.04	0.69	17.23	19.22
512	5.19	0.69	17.51	19.12

Blackmann Window Outputs

N	Signal Amplitude A_S (in V)	Noise Amplitude A_N (in V)	SNR $20*\log_{10}(A_S/A_N)$ (in dB)	SNR using MATLAB Function (in dB)
8	0.96	0.18	14.60	17.55
64	5.04	0.69	17.23	19.20
512	5.19	0.69	17.51	19.12

Results:

Thus, the effect of using different windows in order to create practically realisable filters was observed. The frequency response of the filters was plotted, and then noisy signals were passed through the filters. The output showed that the stop-band components were attenuated in the output, indicating that the filter works as desired.

Since the input and output signals were sampled at 6 kHz, the cut-off frequency of the filters was $\frac{0.8 \cdot 3000}{\pi} = 763.9kHz$. This is why the 500 Hz signal was allowed to pass whereas the 1.5 kHz signal was attenuated.

Discussion:

The time-domain representation of a low-pass filter is an impulse. An ideal impulse extends to infinite time since we want a sharp frequency cut-off. However, an infinite impulse is not practical, so we restrict it in the time domain by multiplying the infinite impulse with a windowing function. The introduction of a restriction in time-domain leads to deviation from the ideal frequency response of a filter, due to the Gibb's Phenomenon. Depending on the windowing function, we observed that the frequency response of the filter had a finite roll-off and non-zero transition width, and ripples in the pass-band as well as the stop-band. The transition width, peak of the side-lobes and the max attenuation was noted. As N (length of the filter) increased, the transition width decreased, and it approached the ideal response. Similarly, as N increased, the maximum stop-band attenuation decreased, which is also desirable. Also, the peak of the side lobe decreased as the length of the filter increased. For triangular windowing with N=512, no proper ripples were observed in the stop band due to which the peak of the side lobe and transition width could not be calculated. For calculating these values, the responses were plotted in dB scale, but since the representation in absolute units looks much better, those absolute value plots are pictured in the report.

For testing the filters that were designed, an input signal was generated with components in both, the pass-band as well as the stop-band. The input signal was passed through the filter and the

input and output signals were plotted in the time-domain as well as frequency domain. We see clearly that the higher frequency component has been attenuated in the output, indicating that our lowpass filter works as desired. Some noise was also added to the input signal and it was filtered as well. For the output signals, the noise and signal amplitudes were noted and the SNR was obtained. There is a difference between the SNR obtained from the amplitudes and the SNR obtained using the MATLAB function. This occurs since the actual power of the noise signal is lesser than the value obtained from squaring the maximum amplitude of the noise signal. This means the actual power of the noise cannot be estimated by squaring the maximum amplitude, but rather the sum of squares of the samples must be taken. This approach is taken by the MATLAB function and so the reading is much more accurate.

With this simulation, we see the effect different windows have on the filter. The Blackmann window has the best attenuation in the stop-band, whereas the Hanning window results in the shortest transition width. So, depending on the requirements, we can choose an appropriate window so that we get the desired behaviour.

Appendix:

1. Code for generating and plotting windowed filters

```
clc
clear all
close all

N = 512; %declaring window length
k = floor((N-1)/2);
n = 0:1:(N-1);

wc = 0.8; %normalized cut-off frequency in rads
w = -pi:1/2000:pi;

hd = zeros(1, N); %initialising desired impulse response
for ii = 1:N
   if ii == k
      hd(ii) = wc/pi;
   else
      hd(ii) = sin(wc*(ii-k))/(pi*(ii-k));
   end
end

%defining window functions
rectangular = ones(1, N);
triangular = 1 - 2*abs(n-(N-1)/2)/(N-1);
hanning = 0.5 - 0.5*cos((2*pi/(N-1))*n);
hamming = 0.54 - 0.46*cos((2*pi/(N-1))*n);
blackmann = 0.42 - 0.5*cos((2*pi/(N-1))*n) + 0.08*cos((4*pi/(N-1))*n);
impulse_mat = [hd; rectangular; triangular; hanning; hamming; blackmann];

%plotting ideal impulse and window functions
figure()
sgtitle("Impulse responses for N="+num2str(N));
for ii = 1:6
   if ii == 1
      name = "Ideal Impulse";
   elseif ii == 2
      name = "Rectangular Window";
   elseif ii == 3
      name = "Triangular Window";
   elseif ii == 4
      name = "Hanning Window";
   elseif ii == 5
      name = "Hamming Window";
   else
      name = "Blackmann Window";
   end
   subplot(3,2,ii);
   plot(n,impulse_mat(ii,:));
   xlim([0 N]);
   grid on
   xlabel('Time (or n)'); ylabel('x(n)'); title(name);
end
```

```
%multiplying windows with ideal impulse in time-domain
h_rect = hd.*rectangular;
h_trig = hd.*triangular;
h_hann = hd.*hanning;
h_hamm = hd.*hamming;
h_black = hd.*blackmann;

h_mat = [h_rect; h_trig; h_hann; h_hamm; h_black];

%plotting all frequency responses
for ii=1:5
   if ii == 1
      name = "Rectangular";
   elseif ii == 2
      name = "Triangular";
   elseif ii == 3
      name = "Hanning";
   elseif ii == 4
      name = "Hamming";
   else
      name = "Blackmann";
   end

   [H,W] = freqz(h_mat(ii,:), 1, w);
   figure()
   subplot(2,1,1)
   plot(W, abs(H));
   grid on
   xlabel('Normalized Frequency in rads/sample');
   ylabel('abs(H(w))');
   title("Frequency Response (Magnitude) for "+name+" Window when N=" + N);
   subplot(2,1,2)
   plot(W, angle(H));
   grid on
   xlabel('Normalized Frequency in rads');
   ylabel('angle(H(w)) in radians');
   title("Frequency Response (Phase) for "+name+" Window when N=" + N);
end
```

2. Code for testing the different filters

```
clc
clear all
close all

N = 512; %declaring length of filter
k = floor((N-1)/2);
n = 0:1:(N-1);

wc = 0.8; %declaring normalised cut-off frequency in rads
w = -pi:1/2000:pi;

hd = zeros(1, N); %initialising ideal impulse response
for ii = 1:N
   if ii == k
```

```
        hd(ii) = wc/pi;
    else
        hd(ii) = sin(wc*(ii-k))/(pi*(ii-k));
    end
end

%declaring window functions
rectangular = ones(1, N);
triangular = 1 - 2*(n-(N-1)/2)/(N-1);
hanning = 0.5 - 0.5*cos((2*pi/(N-1))*n);
hamming = 0.54 - 0.46*cos((2*pi/(N-1))*n);
blackmann = 0.42 - 0.5*cos((2*pi/(N-1))*n) + 0.08*cos((4*pi/(N-1))*n);

%multiplication in time domain to get windowed filter
h_rect = hd.*rectangular;
h_trig = hd.*triangular;
h_hann = hd.*hanning;
h_hamm = hd.*hamming;
h_black = hd.*blackmann;

%%%% Generating an input signal and adding noise %%%%
f_pass = 500;
f_stop = 1500;
fs = 6000; %sampling frequency
t = 0:1/fs:(3*N-1)/fs;
rng('default') %to ensure same output from rand every time
noise = rand(1, 3*N);
x = 5*sin(2*pi*f_pass*t) + 5*sin(2*pi*f_stop*t);
add_noise = (max(x)/10)*noise/abs(max(noise)); %to control SNR
noisy_x = x + add_noise;
f_eq = -3000:2000/N:3000-2000/N;

h_matrix = [h_rect; h_trig; h_hann; h_hamm; h_black];

%%% Filtering the input signal through the various filters %%%
for ii=1:5
    if ii == 1
        name = "Rectangular";
    elseif ii == 2
        name = "Triangular";
    elseif ii == 3
        name = "Hanning";
    elseif ii == 4
        name = "Hamming";
    else
        name = "Blackmann";
    end
    y = filtfilt(h_matrix(ii,:), 1, x);
    y_n = filtfilt(h_matrix(ii,:), 1, noisy_x);

    %plotting time-domain response
    figure()
    sgtitle(name+" Window Time Domain");
    subplot(221);
    %plotting limited samples for clarity
    plot(t(1:floor(15*fs/f_stop)),x(1:floor(15*fs/f_stop)));
    grid on
    xlabel('Time in seconds'); ylabel('x(t)'); title('Input Signal');
```

```
    subplot(222);
    plot(t(1:floor(15*fs/f_stop)),noisy_x(1:floor(15*fs/f_stop)));
    grid on
    xlabel('Time in seconds'); ylabel('x(t)'); title('Input Signal with Noise');
    subplot(223);
    plot(t(1:floor(15*fs/f_stop)),y(1:floor(15*fs/f_stop)));
    grid on
    xlabel('Time in seconds'); ylabel('y(t)'); title('Output Signal');
    subplot(224);
    plot(t(1:floor(15*fs/f_stop)),y_n(1:floor(15*fs/f_stop)));
    grid on
    xlabel('Time in seconds'); ylabel('y(t)'); title('Output Signal with Noise');

    %plotting frequency domain response
    figure()
    sgtitle(name+" Window Frequency Domain");
    subplot(2,2,1)
    plot(f_eq, 20*log10(abs(fftshift(fft(x)))));
    xlabel('Frequency in Hz'); ylabel('Input Spectrum (in dB)'); title('Input Spectrum of Signal
without Noise')
    grid on;
    subplot(2,2,2)
    plot(f_eq, 20*log10(abs(fftshift(fft(noisy_x)))));
    xlabel('Frequency in Hz'); ylabel('Input Spectrum (in dB)'); title('Input Spectrum of Signal with
Noise')
    grid on;
    subplot(2,2,3)
    plot(f_eq, 20*log10(abs(fftshift(fft(y)))));
    xlabel('Frequency in Hz'); ylabel('Output Spectrum (in dB)'); title('Output Spectrum of Signal
without Noise')
    grid on;
    subplot(2,2,4)
    plot(f_eq, 20*log10(abs(fftshift(fft(y_n)))));
    xlabel('Frequency in Hz'); ylabel('Output Spectrum (in dB)'); title('Output Spectrum of Signal
with Noise')
    grid on;

    %printing amplitudes and SNR values
    max_out = max(y);
    max_out_noise = max(y_n-y);
    calc_snr = 20*log10(max_out/max_out_noise);
    fprintf("%0.2f %0.2f %0.2f \n",max_out, max_out_noise, calc_snr)
    disp("Input SNR(in dB) for "+name+" window = "+snr(x, noisy_x-x))
    disp("Output SNR(in dB) for "+name+" window = "+snr(y, y_n-y))
end
```