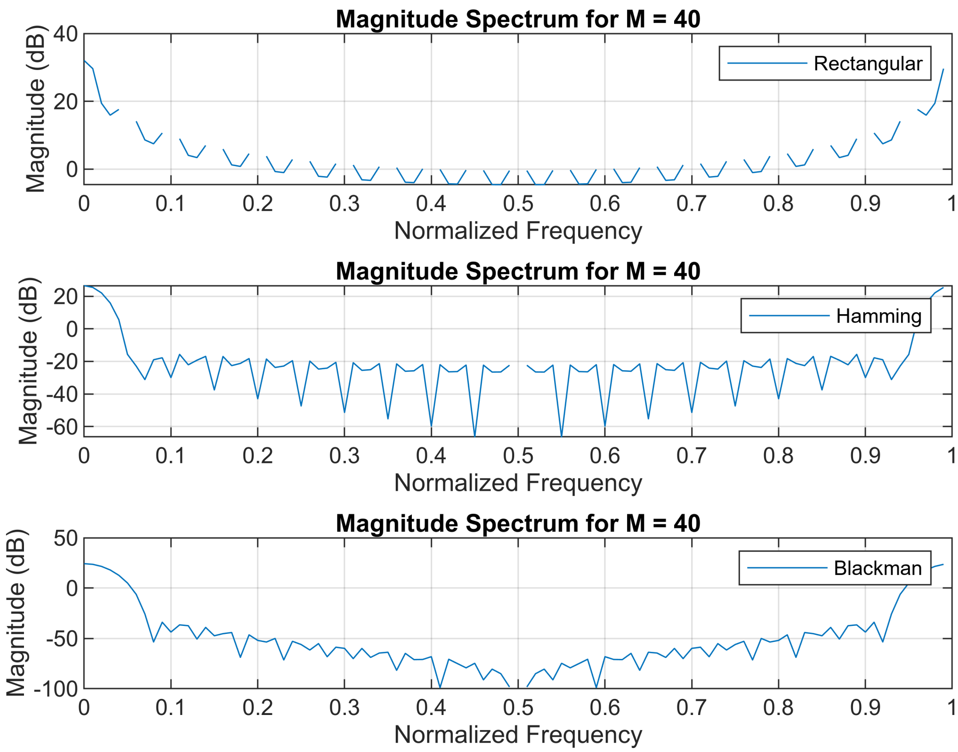
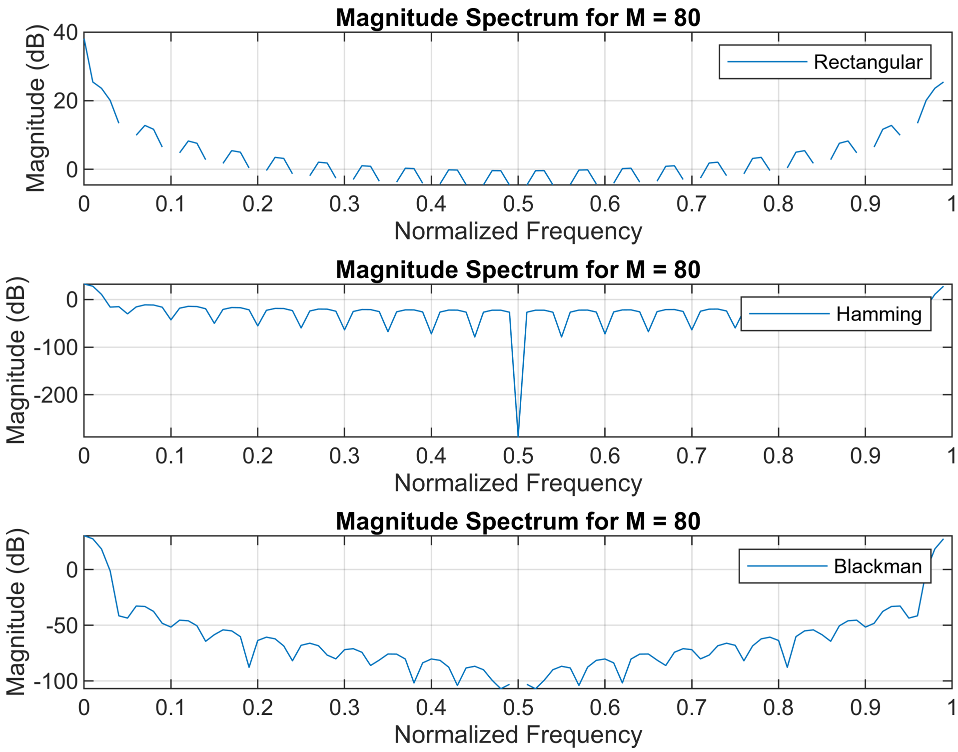
PART B

```
M_values = [40, 80];

for M = M_values
    figure;
    subplot(3,1,1);
    w_rec = rec(M);
    plotMagnitudeSpectrum(w_rec, M);
    legend('Rectangular');

    subplot(3,1,2);
    w_ham = ham(M);
    plotMagnitudeSpectrum(w_ham, M);
    legend('Hamming');

    subplot(3,1,3);
    w_bla = bla(M);
    plotMagnitudeSpectrum(w_bla, M);
    legend('Blackman');
end
```



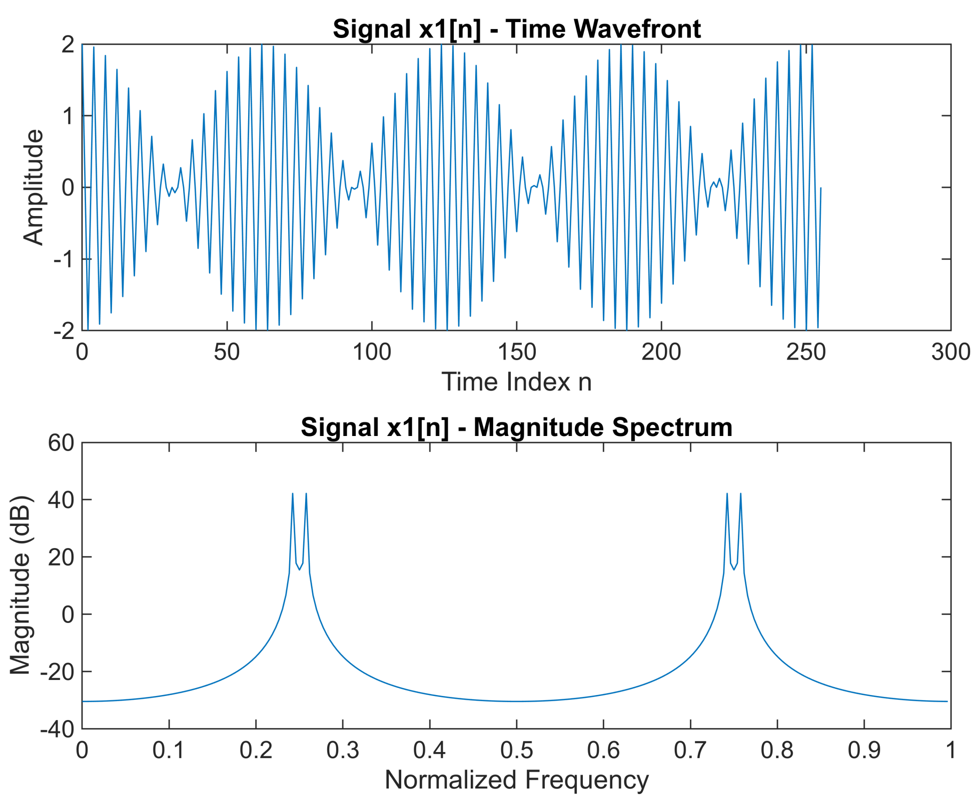



```
% Spectra Comparison

% The main lobe width and side lobe level are important characteristics of window
functions.
% They determine the resolution and leakage of the spectral analysis.

% Main Lobe Width:
% The main lobe width is inversely related to the window length. A larger window
length (M) results
% in a narrower main lobe, which improves frequency resolution because it allows
for better
% discrimination between closely spaced frequency components.

% Side Lobe Level:
% The side lobe level refers to the peaks that occur in the spectrum outside the
main lobe.
% Lower side lobes are generally desired to reduce leakage, which is the spreading
of signal energy
% into frequencies outside the main lobe. High side lobes can mask or distort the
presence of small
% signals close to larger ones.

% Effect of Window Length:
```

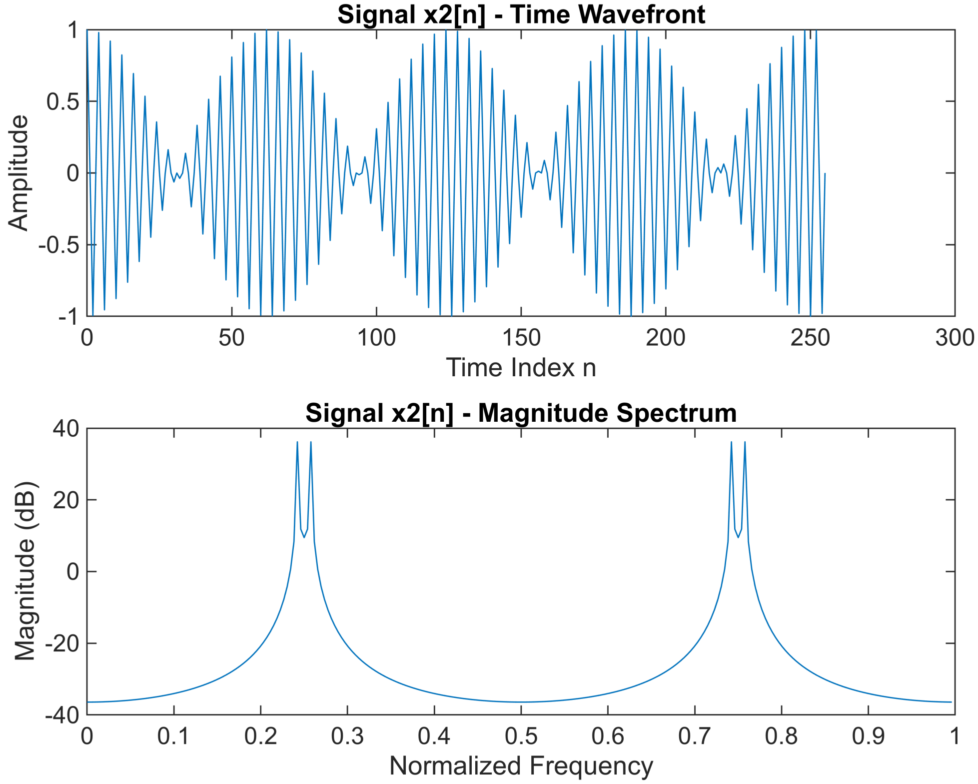
```
% Increasing the window length (M) generally results in lower side lobes, which
means less leakage
% and better signal representation. However, this comes at the cost of
computational efficiency
% and requires more data points.

% In summary, a longer window length (M) improves frequency resolution and reduces
side lobe levels,
% but requires more computational resources and data points.
```

PART C

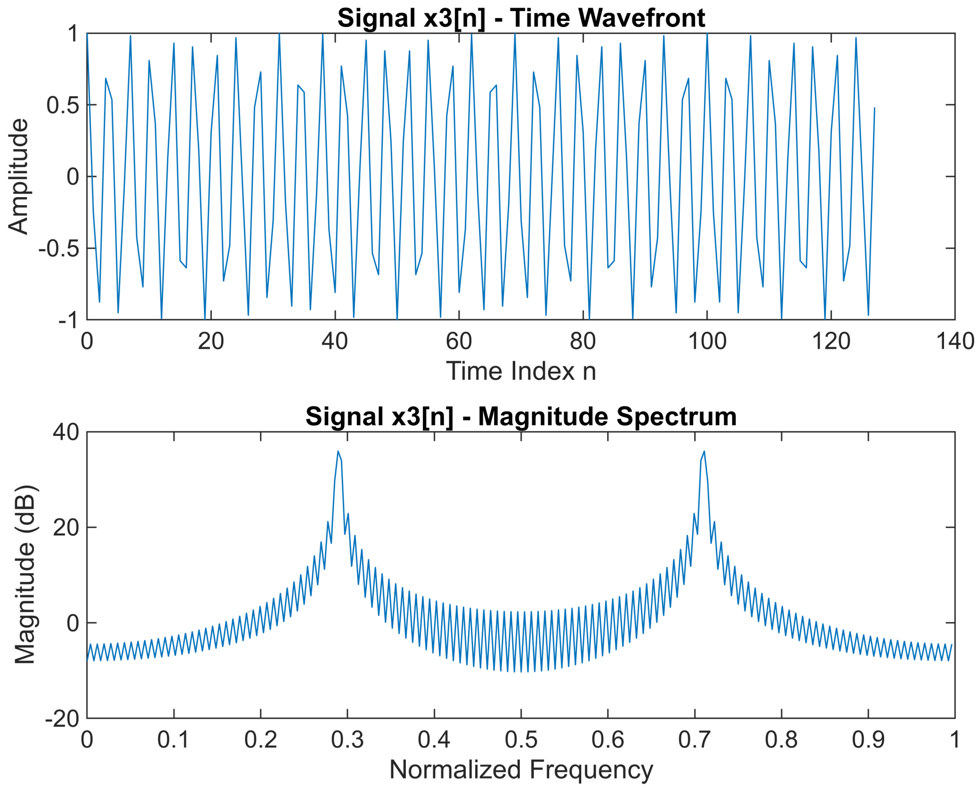
```
n_256 = 0:255;
n_128 = 0:127;
x1 = cos(2 * pi * n_256 * 0.242) + cos(2 * pi * n_256 * 0.258);
x2 = cos(2 * pi * n_256 * 0.25) .* cos(2 * pi * n_256 * 0.008);
x3 = cos(2 * pi * n_128 * 0.29);

figure;
plotSignal(n_256, x1, 'Signal x1[n]');
```




```
figure;
plotSignal(n_256, x2, 'Signal x2[n]');
```




```
figure;
plotSignal(n_128, x3, 'Signal x3[n]');
```

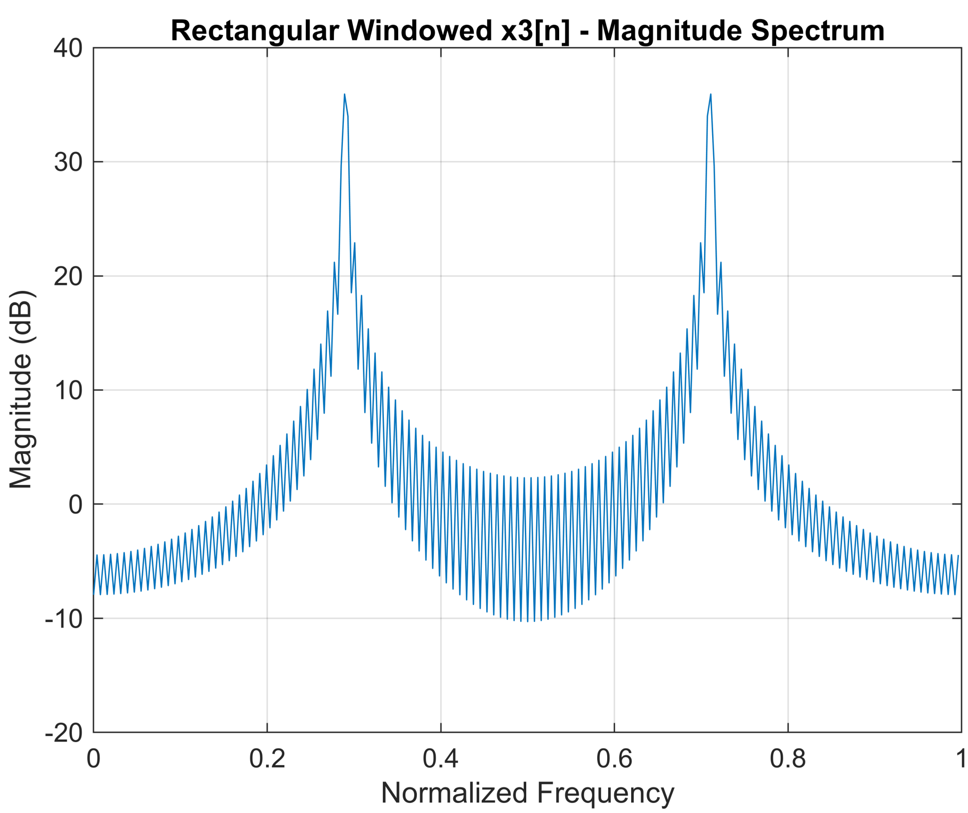



```
% Comparison of x1[n] and x2[n]

% Time Wavefront:
% x1[n] = cos(2*pi*n*0.242) + cos(2*pi*n*0.258)
% This signal is a superposition of two cosine waves with frequencies that are
close to each other.
% The time wavefront will exhibit a beat phenomenon due to the interference of
these two frequencies.

% x2[n] = cos(2*pi*n*0.25) * cos(2*pi*n*0.008)
% This signal is a product of two cosine waves, which results in amplitude
modulation.
% The time wavefront will show a cosine wave with a varying amplitude envelope.

% Frequency Content (Magnitude Spectrum):
% The magnitude spectrum of x1[n] will show two distinct peaks at the frequencies
0.242 and 0.258.
% These peaks represent the frequency components of the signal.

% The magnitude spectrum of x2[n] will show a more complex structure due to the
modulation.
% It will have a central peak at the carrier frequency 0.25 and sidebands at
frequencies resulting
% from the sum and difference of the carrier and modulating frequencies.
```

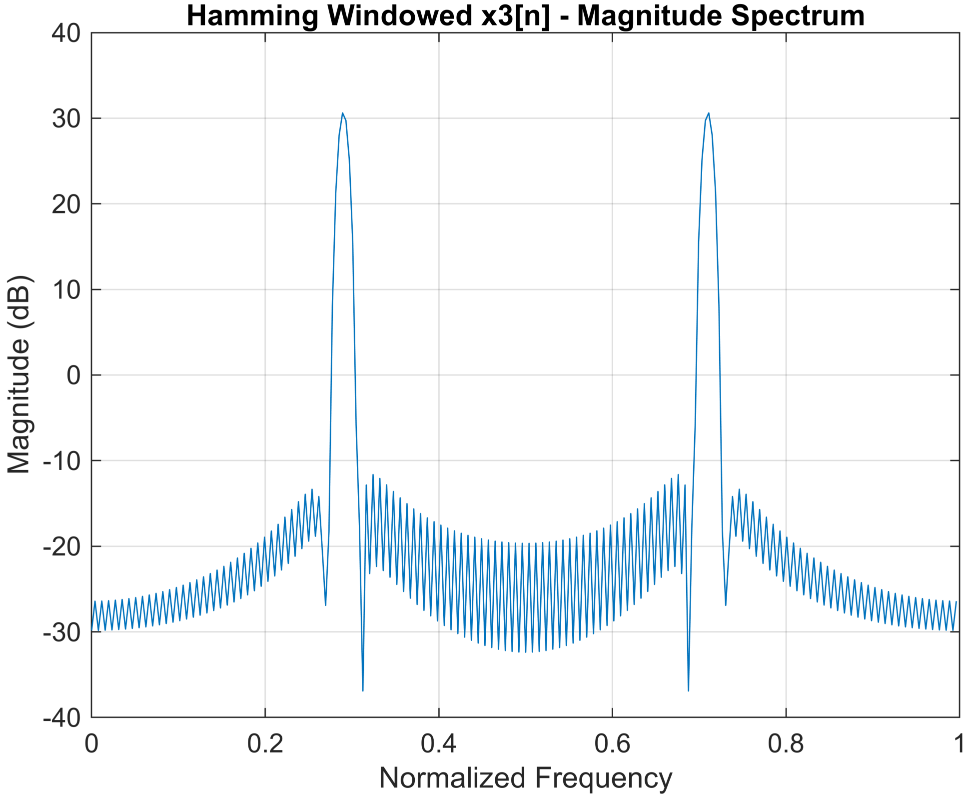
```
% When comparing the magnitude spectra of x1[n] and x2[n], x1[n] will have sharper
peaks due to the
% simple addition of frequencies, while x2[n] will have additional components due
to the modulation
% effect, which spreads the energy into multiple frequencies.

% The main lobe width in the magnitude spectrum is determined by the window length
used in the FFT.
% A longer window length will result in narrower main lobes, which provides better
frequency resolution.
% Side lobes are also affected by the window type; for example, a Hamming window
will reduce the side
% lobe levels compared to a rectangular window, leading to less spectral leakage.
```

PART D

```
n_128 = 0:127;
x3 = cos(2 * pi * n_128 * 0.29);
w_rec = rec(128);
w_ham = ham(128);
x3_rec = x3 .* w_rec';
x3_ham = x3 .* w_ham';

figure;
plotMagnitudeSpectrumWindowed(x3_rec, w_rec, 'Rectangular Windowed x3[n]');
```



```
figure;
plotMagnitudeSpectrumWindowed(x3_ham, w_ham, 'Hamming Windowed x3[n]');
```

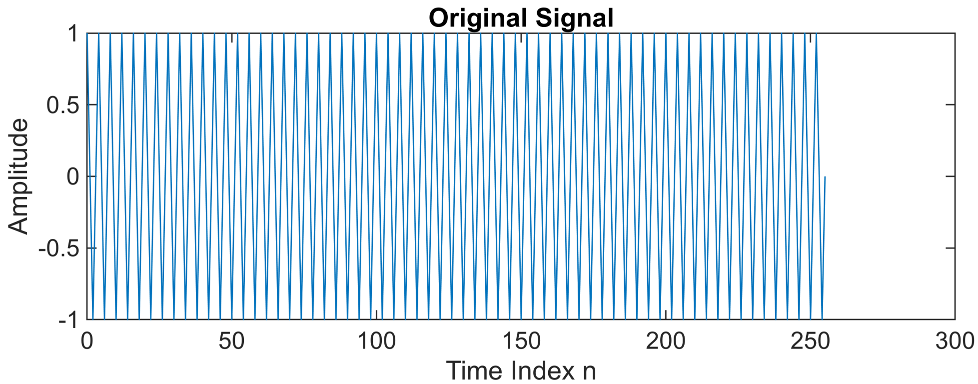


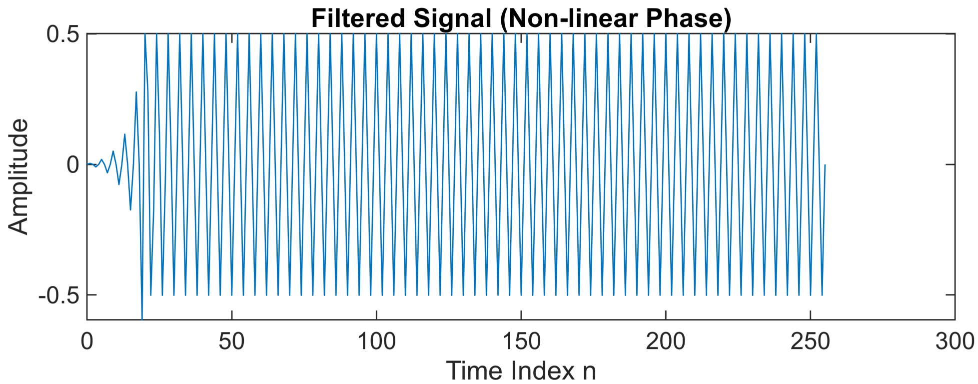

```
% Comparison of Windowing Effects: Rectangular vs. Hamming Window

% Time Domain:
% Rectangular window: No modification within the span, abrupt cutoff outside.
% Hamming window: Smooth tapering at the edges, reducing spectral leakage.

% Frequency Domain:
% Rectangular window: Narrower main lobe (better resolution), higher side lobes
(potential masking).
% Hamming window: Wider main lobe (slightly reduced resolution), lower side lobes
(less leakage).

% If x3[n] were windowed with a Hamming window:
% The magnitude spectrum would have a wider main lobe and lower side lobes.
% This results in a smoother spectrum and a clearer view of the signal's frequency
content.
```

PART E

```
% Advantages of FIR filters:
```

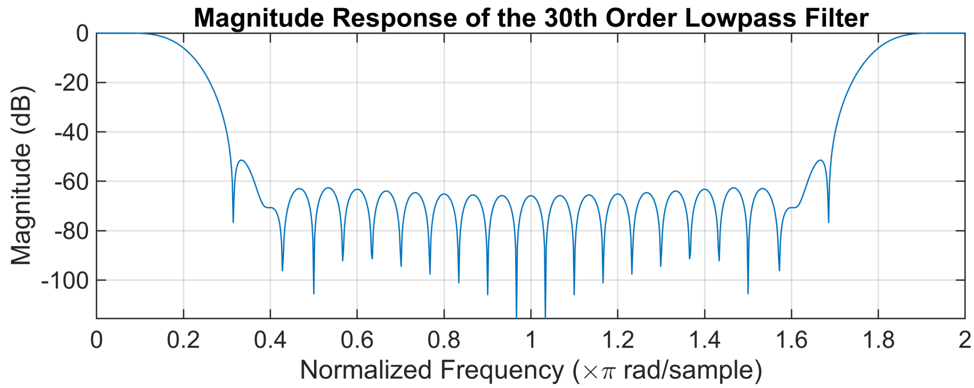
```
% 1. They are inherently stable since they do not have feedback loops.
% 2. They can have exactly linear phase response, which means no phase distortion.
% 3. They can be designed to have a symmetric impulse response, which is desired in
many applications.

% What is a 'linear phase filter'?
% A linear phase filter is a filter whose phase response is a linear function of
frequency.
% This means that all frequency components of the input signal are delayed by the
same amount of time,
% preserving the wave shape of signals within the passband.

% Example of non-linear phase filter effect:
% If a filter is not linear phase, different frequency components are delayed by
different amounts,
% causing phase distortion. This can be particularly problematic in applications
like data communications
% or audio processing, where maintaining the integrity of the signal's phase is
crucial.

n = 0:255;
x = cos(2 * pi * n * 0.25);
b = fir1(40, 0.5, 'high', chebwin(41,30));
y = filter(b, 1, x);

figure;
subplot(2,1,1);
plot(n, x);
title('Original Signal');
xlabel('Time Index n');
ylabel('Amplitude');

subplot(2,1,2);
plot(n, y);
title('Filtered Signal (Non-linear Phase)');
xlabel('Time Index n');
ylabel('Amplitude');
```

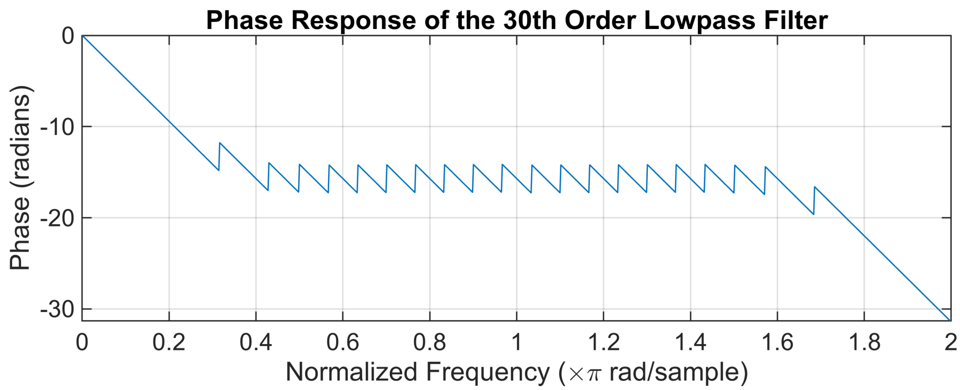




PART F

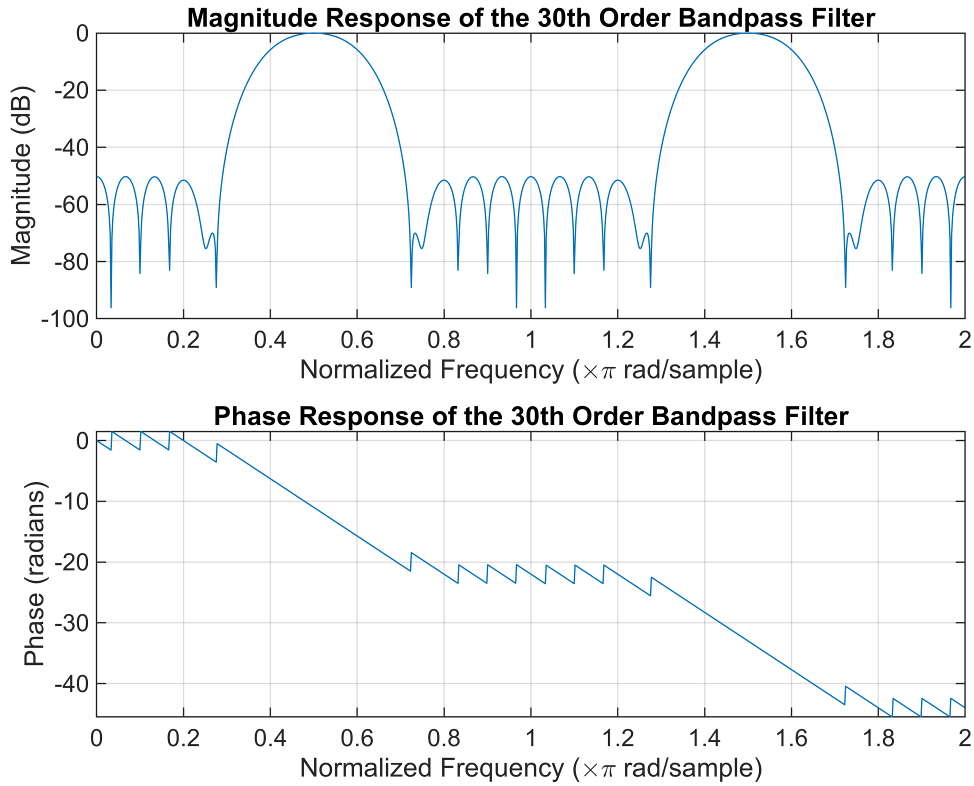
```
N = 30;
wc = 0.2 * pi;
b = fir1(N, wc/(pi), 'low');
[H, f] = freqz(b, 1, 1024, 'whole');
H_dB = 20 * log10(abs(H));
H_phase = unwrap(angle(H));

figure;
subplot(2,1,1);
plot(f/pi, H_dB);
title('Magnitude Response of the 30th Order Lowpass Filter');
xlabel('Normalized Frequency (\times\pi rad/sample)');
ylabel('Magnitude (dB)');
grid on;

subplot(2,1,2);
plot(f/pi, H_phase);
title('Phase Response of the 30th Order Lowpass Filter');
xlabel('Normalized Frequency (\times\pi rad/sample)');
ylabel('Phase (radians)');
grid on;
```





PART G

```
N = 30;
wp = [0.4 * pi, 0.6 * pi] / pi;
b = fir1(N, wp, 'bandpass', hamming(N+1));
[H, f] = freqz(b, 1, 1024, 'whole');
H_dB = 20 * log10(abs(H));
H_phase = unwrap(angle(H));

figure;
subplot(2,1,1);
plot(f/pi, H_dB);
title('Magnitude Response of the 30th Order Bandpass Filter');
xlabel('Normalized Frequency (\times\pi rad/sample)');
ylabel('Magnitude (dB)');
grid on;

subplot(2,1,2);
plot(f/pi, H_phase);
title('Phase Response of the 30th Order Bandpass Filter');
xlabel('Normalized Frequency (\times\pi rad/sample)');
ylabel('Phase (radians)');
grid on;
```



FUNCTIONS

```
function w = rec(M)
    w = boxcar(M);
end


function w = ham(M)
    w = hamming(M);
end


function w = bla(M)
    w = blackman(M);
end


function plotMagnitudeSpectrum(w, M)
    W = fft(w, 100);
    W_dB = 20 * log10(abs(W));
    f = (0:99) * 1/100;
    plot(f, W_dB);
    title(['Magnitude Spectrum for M = ', num2str(M)]);
    xlabel('Normalized Frequency');
    ylabel('Magnitude (dB)');
```

```
    grid on;
end


function plotSignal(n, x, titleStr)
    subplot(2,1,1);
    plot(n, x);
    title([titleStr, ' - Time Wavefront']);
    xlabel('Time Index n');
    ylabel('Amplitude');

    subplot(2,1,2);
    X = fft(x, 256);
    X_dB = 20 * log10(abs(X));
    f = (0:255) * 1/256;
    plot(f, X_dB);
    title([titleStr, ' - Magnitude Spectrum']);
    xlabel('Normalized Frequency');
    ylabel('Magnitude (dB)');
end


function plotMagnitudeSpectrumWindowed(x, w, titleStr)
    X = fft(x, 256);
    X_dB = 20 * log10(abs(X));
    f = (0:255) * 1/256;
    plot(f, X_dB);
    title([titleStr, ' - Magnitude Spectrum']);
    xlabel('Normalized Frequency');
    ylabel('Magnitude (dB)');
    grid on;
end
```