Istanbul Technical University

Faculty of Electrical and Electronics Engineering

DIGITAL SIGNAL PROCESSİNG 4TH HOMEWORK (with MATLAB codes)

Name&Surname : Buğrahan Adalı
No : 040210228
Date : 12/01/2025

1 [50 pts] Let $x_1(n) = \delta(n) + 2\delta(n-1) + 2\delta(n-2)$ and $x_2(n) = \delta(n) + 2\delta(n-1) + 3\delta(n-2) + 4\delta(n-3)$.

(a) Compute the linear convolution of these signals $y_\ell(n) = x_1(n) * x_2(n)$. Use can use function `conv_m` (or built in `conv`) to calculate convolution. Plot the result using `stem` function.

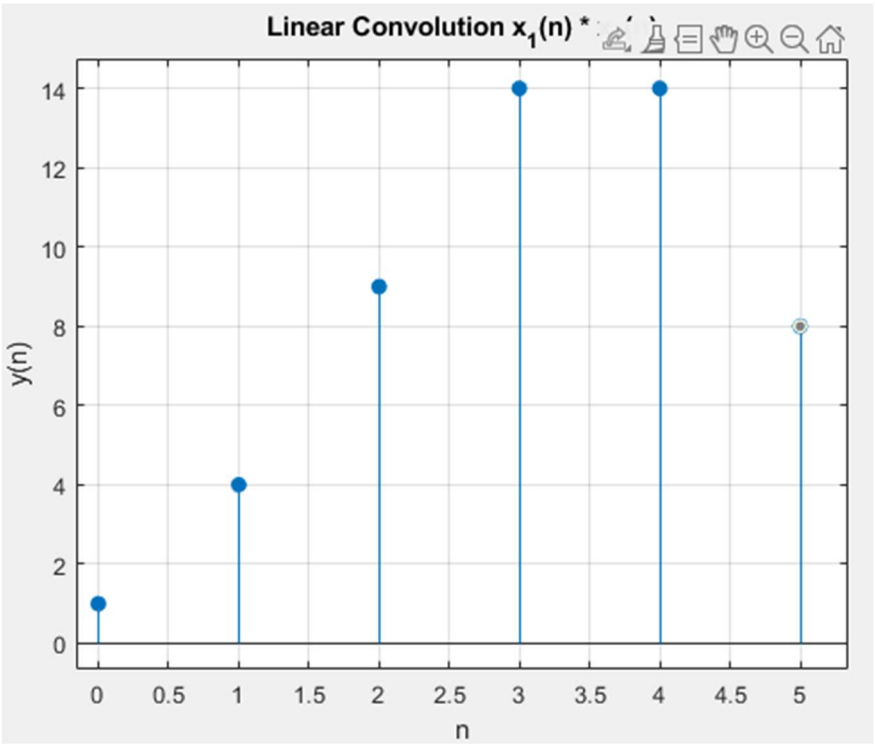
(b) Compute the 4-point circular convolution of these signals $y_4(n) = x_1(n) \textcircled{4} x_2(n)$. Use function `cconv` to calculate circular convolution. Plot the result using `stem` function.

(c) Compute the 8-point circular convolution of these signals $y_8(n) = x_1(n) \textcircled{8} x_2(n)$. Use function `cconv` to calculate circular convolution. Plot the result using `stem` function. item Are $y_\ell(n)$ and $y_4(n)$ equal? Are $y_\ell(n)$ and $y_8(n)$ equal? Discuss.

(d) Compute the 8-point circular convolution of these signals $y_8(n) = x_1(n) \textcircled{8} x_2(n)$ again. This time only use the DFT calculating function `fft`. Did you get the same result as in c) ?

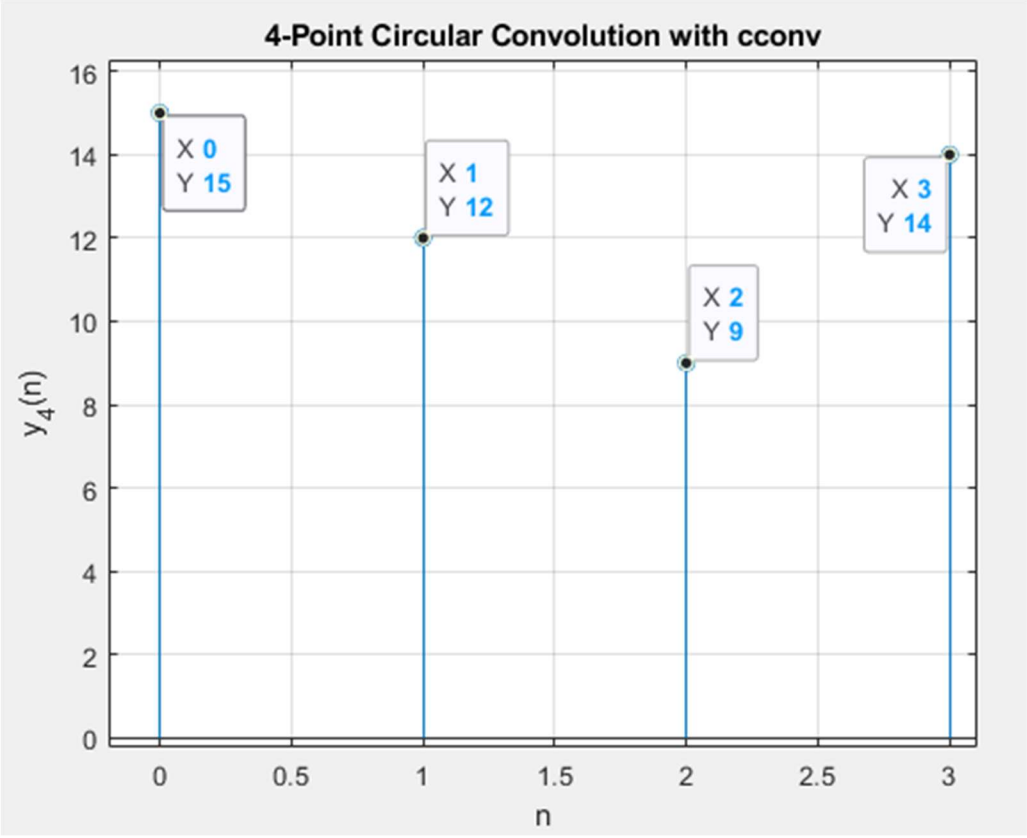
1-a)

```
% (a) Linear Convolution

% 1) Signal definitions:
x1 = [1 2 2];
x2 = [1 2 3 4];

% 2) Linear convolution with conv function:
y_lin = conv(x1, x2);
n_lin = 0 : (length(y_lin)-1);  % Index range for y(n)

% 3) Plot the result:
figure;
stem(n_lin, y_lin, 'filled');
xlabel('n');
ylabel('y(n)', 'Interpreter', 'tex');
title('Linear Convolution x_1(n) * x_2(n)', 'Interpreter', 'tex');
grid on;
```



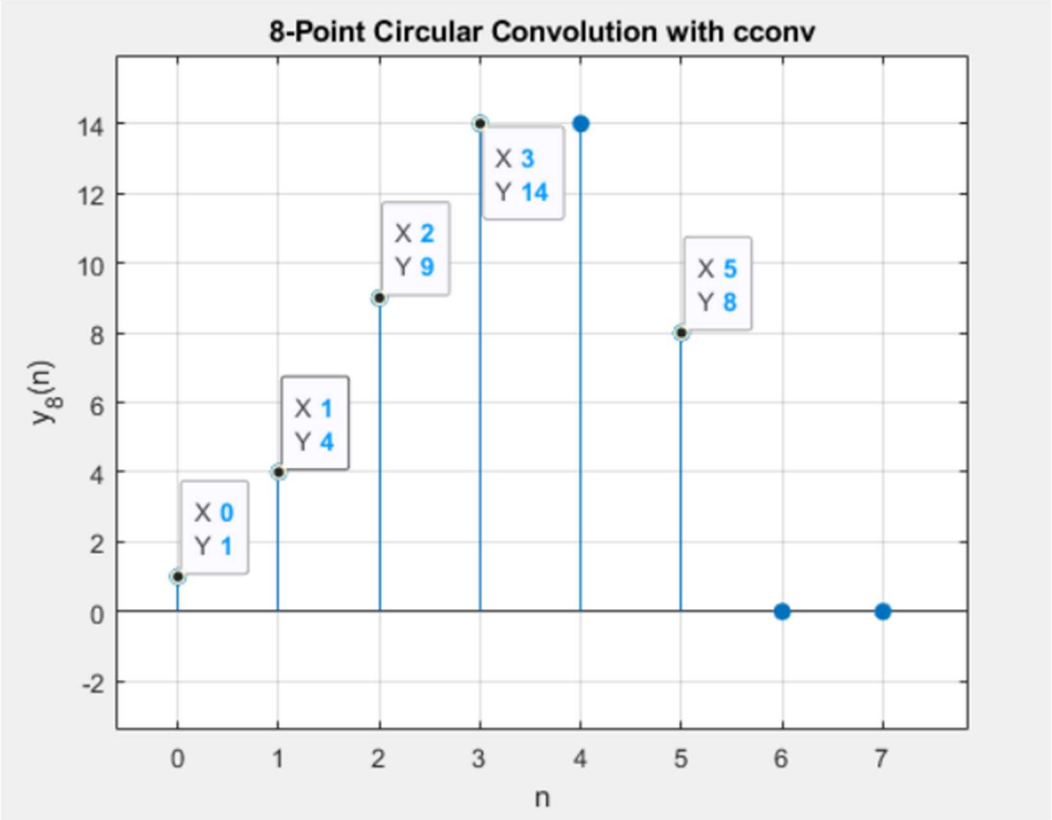
1-b

```
% 4-point circular convolution (Signal Processing Toolbox function cconv)
N4 = 4;
y4 = cconv(x1, x2, N4);

% Sample index vector
n4 = 0 : N4-1;

% Plot the result
figure;
stem(n4, y4, 'filled');
xlabel('n');
ylabel('y_{4}(n)', 'Interpreter', 'tex');
title('4-Point Circular Convolution with cconv', 'Interpreter', 'tex');
grid on;
```



1-c

```
% (c) 8-Point Circular Convolution (cconv ile)
N8 = 8;
y8 = cconv(x1, x2, N8);  % cconv, N8 points circular convolution

% Index vector
n8 = 0:N8-1;

% Plot the result
figure;
stem(n8, y8, 'filled');
xlabel('n');
ylabel('y_{8}(n)', 'Interpreter', 'tex');
title('8-Point Circular Convolution with cconv', 'Interpreter', 'tex');
grid on;
```



Linear convolution is the actual folding of signals, and the length of the result is one less than the sum of the lengths of x_1 and x_2.

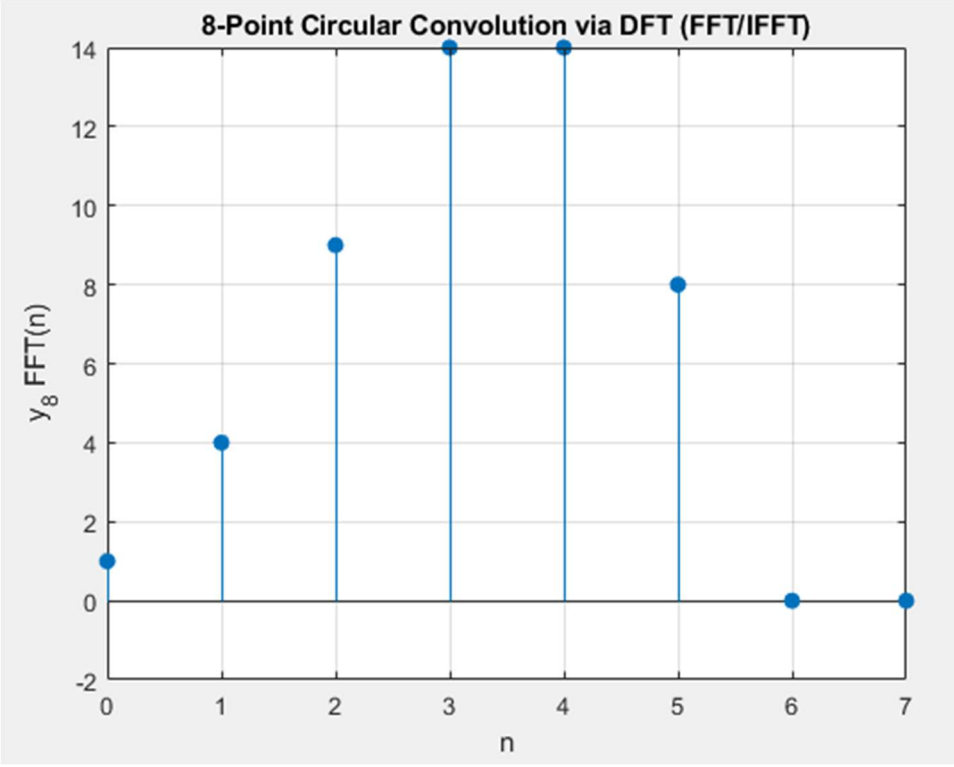
In the 4-point circular convolution, the output is limited to four samples, so parts of the linear convolution overlap, causing aliasing (wrap-around).

However, in the 8-point circular convolution, the result length is eight, which is larger than the linear convolution length (6), so the first six samples match exactly with the linear convolution.

If the circular convolution length is smaller than the total number of samples of the linear convolution, the result “wraps around.” If it is equal to or larger, it matches the linear convolution (at least in the initial portion).

1-d

```
% (d) 8-Point Circular Convolution using FFT

N8 = 8;
% Zero-padding to N8 points
x1_pad8 = [x1, zeros(1, N8 - length(x1))];
x2_pad8 = [x2, zeros(1, N8 - length(x2))];
%  FFT for 8 points
X1 = fft(x1_pad8, N8);
X2 = fft(x2_pad8, N8);
% multiplication in frequency domain
Y_freq = X1 .* X2;
% 4) with IFFT (inverse FFT)
y8_fft = ifft(Y_freq, N8);
% for real-valued signals, take real part
y8_fft = real(y8_fft);

% plot the result
n8 = 0:N8-1;
figure;
stem(n8, y8_fft, 'filled');
xlabel('n');
ylabel('y_8 FFT(n)', 'Interpreter', 'tex');
title('8-Point Circular Convolution via DFT (FFT/IFFT)','Interpreter', 'tex');
grid on;
```



2)

In this question, we focus on designing a lowpass **FIR filter** using the **windowing** method. Four different FIR filters are designed with the **same order** using Hamming, Hanning, Blackman, and Kaiser windows. We then examine the **magnitude** and **phase** responses of these filters and discuss how well each window meets the given passband and stopband specifications.

Sampling Frequency: 20 Hz
Passband Edge Frequency : 2 Hz
Stopband Edge Frequency : 4 Hz
Maximum Passband Ripple : 0.1 dB
Minimum Stopband Attenuation : 40 dB

The goal is to design linear-phase FIR filters.

In the windowing approach ; the ideal impulse response of a lowpass filter (sinc function) is multiplied by a window function (Hamming, Hann, Blackman) to obtain a FIR filter.

The `kaiserord` function, part of MATLAB's Signal Processing Toolbox, uses the desired passband/stopband ripples and the transition frequencies (passband and stopband edges) to estimate the required filter order and the window parameter. This makes designing a Kaiser-windowed filter to meet given specs relatively straightforward.

Parameter Definitions and kasierord

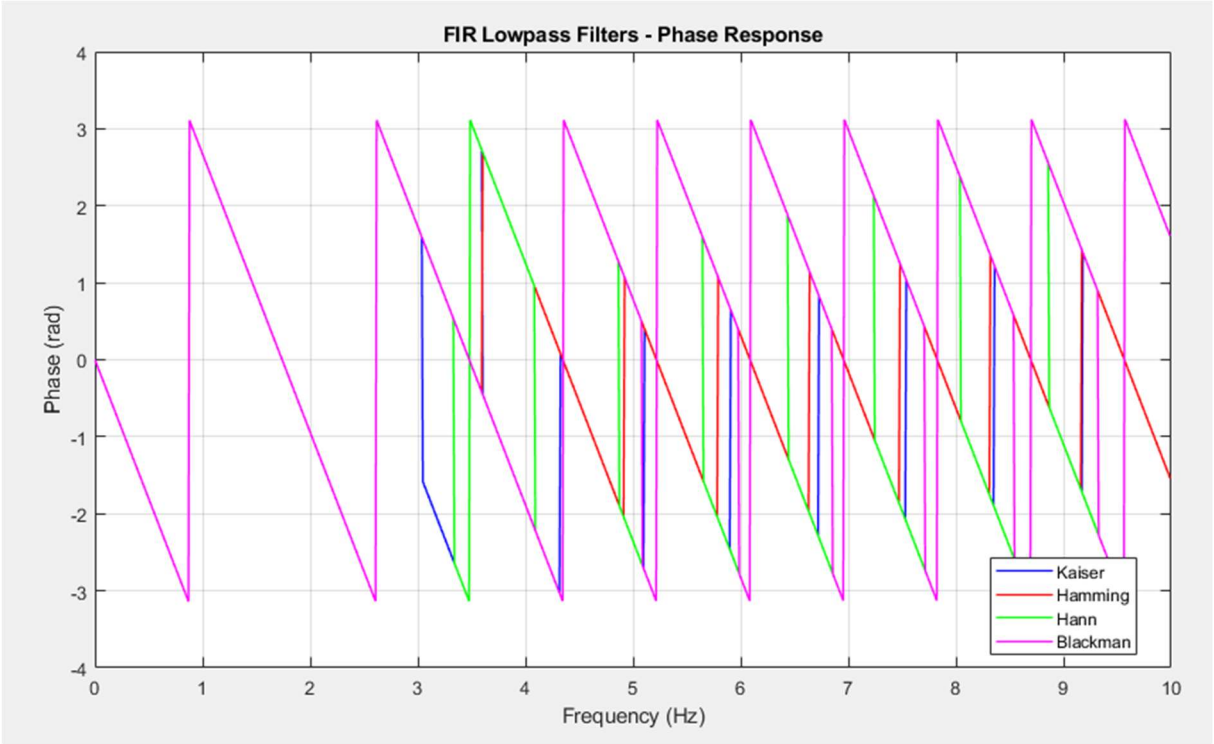
```
% -- Parameters for FIR filter design:
Fs   = 20;     % Sampling frequency (Hz)
Fp   = 2;      % Passband edge frequency (Hz)
Fs1  = 4;      % Stopband edge frequency (Hz)
Ap   = 0.1;    % Passband ripple (dB)
As   = 40;     % Stopband attenuation (dB)

% -- Normalized frequencies:
Wp = Fp / (Fs/2);   % 0.2
Ws = Fs1 / (Fs/2);  % 0.4
% Normalization equation: W = 2*f/Fs
% Wp = 2*f/Fs --> f = Wp*Fs/2
% (why they are normalized? : to be between 0 and 1)

% -- dB --> linear conversion:
Rp = 10^(Ap/20) - 1;    % 0.0389
Rs = 10^(-As/20);       % 0.01

% -- Kaiser window design:
[N, Wn, beta, ftype] = kaiserord([Fp Fs1], [1 0], [Rp Rs], Fs);
% Note: kaiserord may still return a single integer, as needed
% fir1(...) is generally designed with "N" (filter order).
% If we want the order to be even or odd, we can adjust:
% N = N + rem(N,2);  % For example, to make it even.
% -- Kaiser window FIR filter design:
bKaiser = fir1(N, Wp, 'low', kaiser(N+1, beta));
fprintf('Kaiser window FIR filter order: %d\n', N);
```

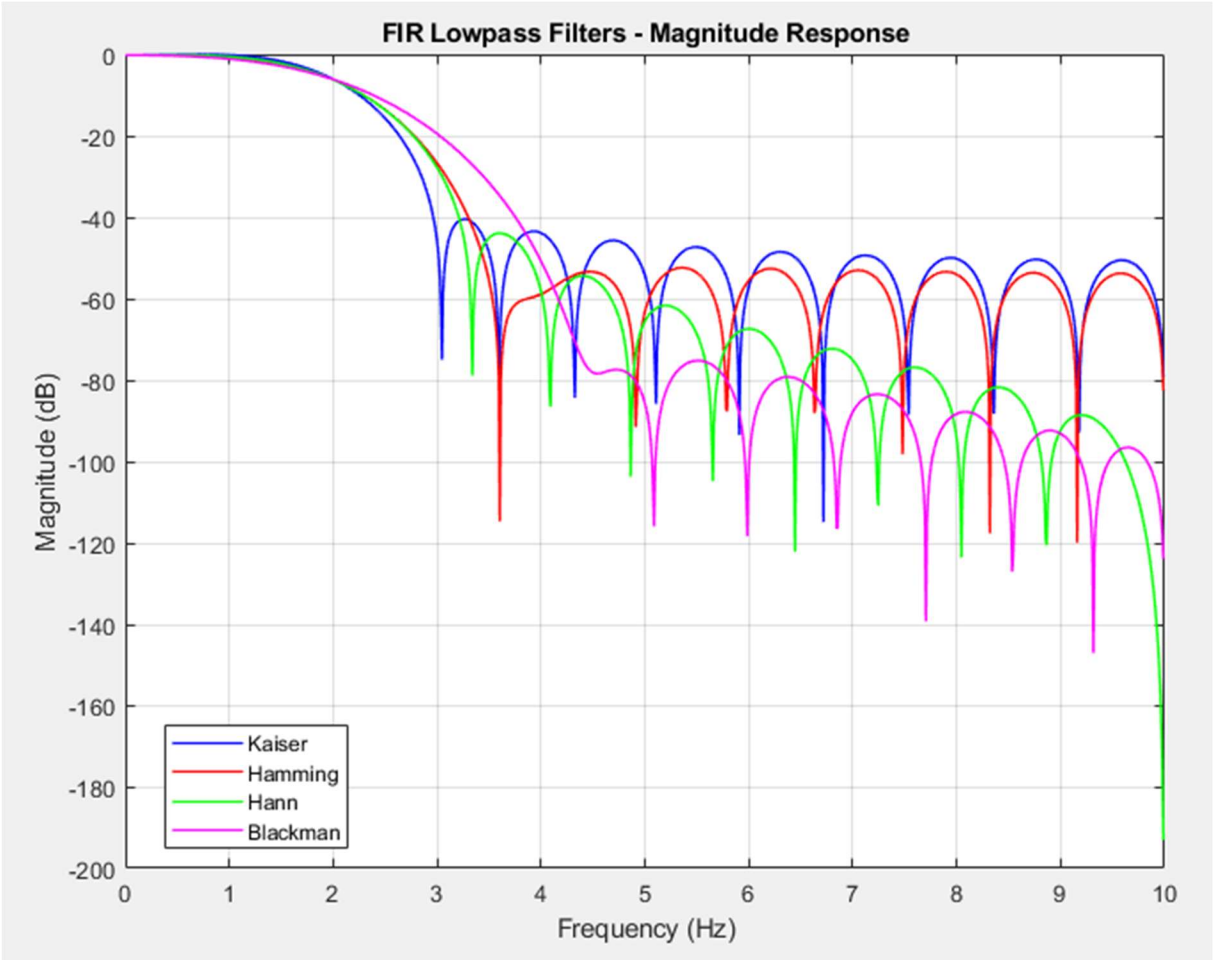
Designing Filters with other Windows

```
% Hamming window:
bHamming = fir1(N, Wp, 'low', hamming(N+1));

% Hanning window:
bHann = fir1(N, Wp, 'low', hanning(N+1));

% Blackman window:
bBlackman = fir1(N, Wp, 'low', blackman(N+1));
```

Plot Magnitude and Phase Responses

```
% Frequency response length:
Nfft = 1024;
[Hk,  W1] = freqz(bKaiser,   1, Nfft, Fs);
[Hhm, W2] = freqz(bHamming,  1, Nfft, Fs);
[Hhn, W3] = freqz(bHann,     1, Nfft, Fs);
[Hbl, W4] = freqz(bBlackman, 1, Nfft, Fs);

% Genlik dB
magK  = 20*log10(abs(Hk));
magHm = 20*log10(abs(Hhm));
magHn = 20*log10(abs(Hhn));
magBl = 20*log10(abs(Hbl));

% Faz (radyan)
phK  = angle(Hk);
phHm = angle(Hhm);
phHn = angle(Hhn);
phBl = angle(Hbl);

figure;
% subplot(2,1,1)
plot(W1, magK, 'b',  W2, magHm, 'r', ...
     W3, magHn, 'g', W4, magBl, 'm','LineWidth',1)
grid on;
xlabel('Frequency (Hz)');
ylabel('Magnitude (dB)');
title('FIR Lowpass Filters - Magnitude Response');
legend('Kaiser','Hamming','Hann','Blackman','Location','Best');

figure;
% subplot(2,1,2)
plot(W1, phK, 'b',  W2, phHm, 'r', ...
     W3, phHn, 'g', W4, phBl, 'm','LineWidth',1)
grid on;
xlabel('Frequency (Hz)');
ylabel('Phase (rad)');
title('FIR Lowpass Filters - Phase Response');
legend('Kaiser','Hamming','Hann','Blackman','Location','Best');
```


FIR Lowpass Filters - Phase Response
Phase (rad)
Frequency (Hz)
Kaiser
Hamming
Hann
Blackman


FIR Lowpass Filters - Magnitude Response
Magnitude (dB)
Frequency (Hz)
Kaiser
Hamming
Hann
Blackman

Results for 2nd question:

Kaiser window:

- Using **kaiserord** determines the filter order N and β based on the desired ripples (Ap=0.1 dB, As=40 dB).
- The resulting filter typically meets the passband ripple (~0.1 dB) and stopband attenuation (40 dB) criteria.

Hamming, Hanning, Blackman Windows:

- These windows all have fixed side lobe and main lobe properties.
- When forced to use the same filter order N (found from Kaiser's design), they might not precisely meet the 0.1 dB passband ripple or 40 dB stopband attenuation requirements.
- For example,
  - Hamming may provide a decent main lobe width but not quite reach 40 dB attenuation.
  - Hann may have insufficient stopband attenuation or more passband ripple.
  - Blackman often has higher stopband attenuation but a wider transition band.

Plots:

- By examining the magnitude plots, you see that the Kaiser-based filter meets the specs more closely around 2 Hz (passband) and 4 Hz (stopband).
- The other windows can have slightly higher passband ripple or less attenuation in the stopband.