EE 3410: DSP: Semester Project 2
FIR Filter Design and Analysis

Goals:

- To design an N-tap FIR filter to remove unwanted signal components.
- To test the designed FIR filter on an arbitrary DT signal.
- To analyze the effect of changing the number of FIR filter tap points (N) on the ability of the filter to remove unwanted signal components.

Note: FIR filter discussion can be found in lectures 32 and 33.

Restrictions:

- All work must be done in MATLAB.
- All spectral analysis must be done using the programmed mathematical definition of DFT and IDFT. You cannot use the MATLAB in-built FFT and DFT functions.
- You cannot use the MATLAB in-built FIR filter generation tool, or any related in-built function, to create your FIR filter. You cannot use any online, or any other software package, tool to generate your FIR filter.

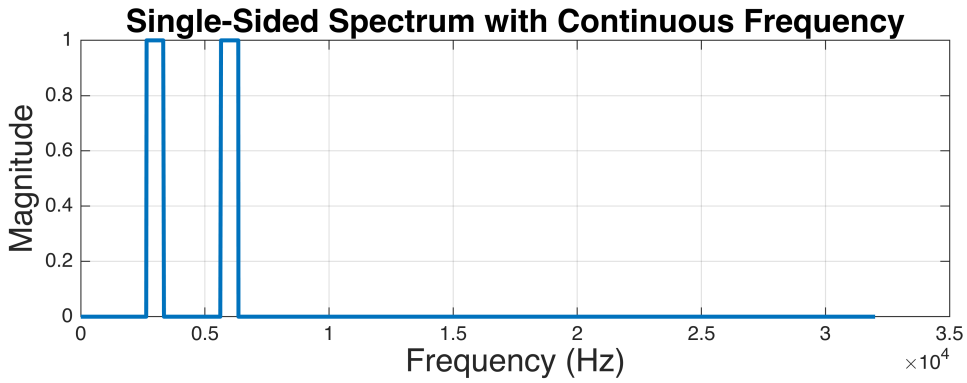
Specifications: Please refer to the table below for the assigned specifications for a **FIR Band Pass Filter (BPF).**

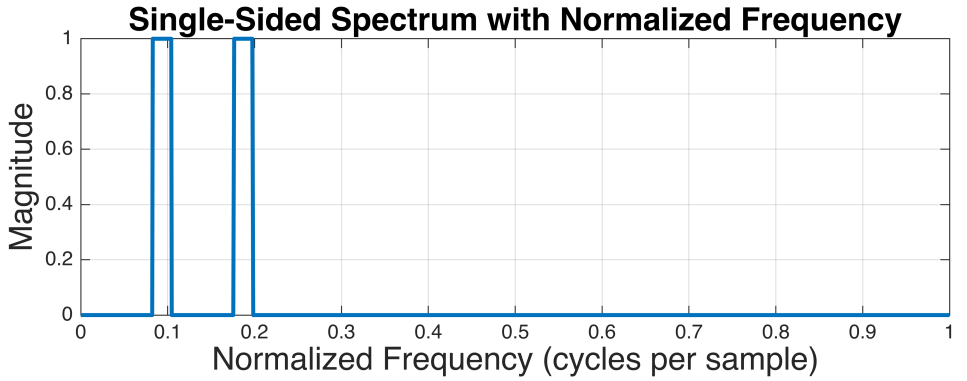
Name	f_C (Center Frequency) (Hz)	Bandwidth (Hz)	Sampling Frequency (Hz)
Trenton Cathcart	3000	700	32000

Single-sided spectrum, of the ideal BPF, versus normalized discrete frequency and continuous frequency in a single figure, but as subplots.

```
clc;
clear all;
close all;

% Define parameters
Fs = 32000;     % Sampling frequency
N1 = 32000;     % Number of samples
F = ((0:1:Fs-1)/N1);  % Normalized frequency range
F_range = F*Fs;       % Continuous frequency range
```

```
fc = 3000;              % Center frequency
BW = 700/Fs;            % Bandwidth

% Create the bandpass filters
% For the single sided continuous frequency spectrum, we center the filter
at fc and 2*fc.
BPFilter_Single_Continuous = (rect(F - (fc/Fs), BW/2) + rect(F - 2*(fc/Fs),
BW/2));
% For the single sided normalized frequency spectrum, we center the filter
at fc and 2*fc.
BPFilter_Single_Normalized = (rect(F - (fc/Fs), BW/2) + rect(F - 2*(fc/Fs),
BW/2));

% Plot the continuous frequency spectrum
subplot(2,1,1)
plot(F_range, BPFilter_Single_Continuous, 'LineWidth', 2);
title('Single-Sided Spectrum with Continuous Frequency', 'FontSize', 16);
xlabel('Frequency (Hz)', 'FontSize', 16);
ylabel('Magnitude', 'FontSize', 16);
grid on;

% Plot the normalized frequency spectrum
subplot(2,1,2)
plot(F, BPFilter_Single_Normalized, 'LineWidth', 2);
title('Single-Sided Spectrum with Normalized Frequency', 'FontSize', 16);
xlabel('Normalized Frequency (cycles per sample)', 'FontSize', 16);
ylabel('Magnitude', 'FontSize', 16);
grid on;
```

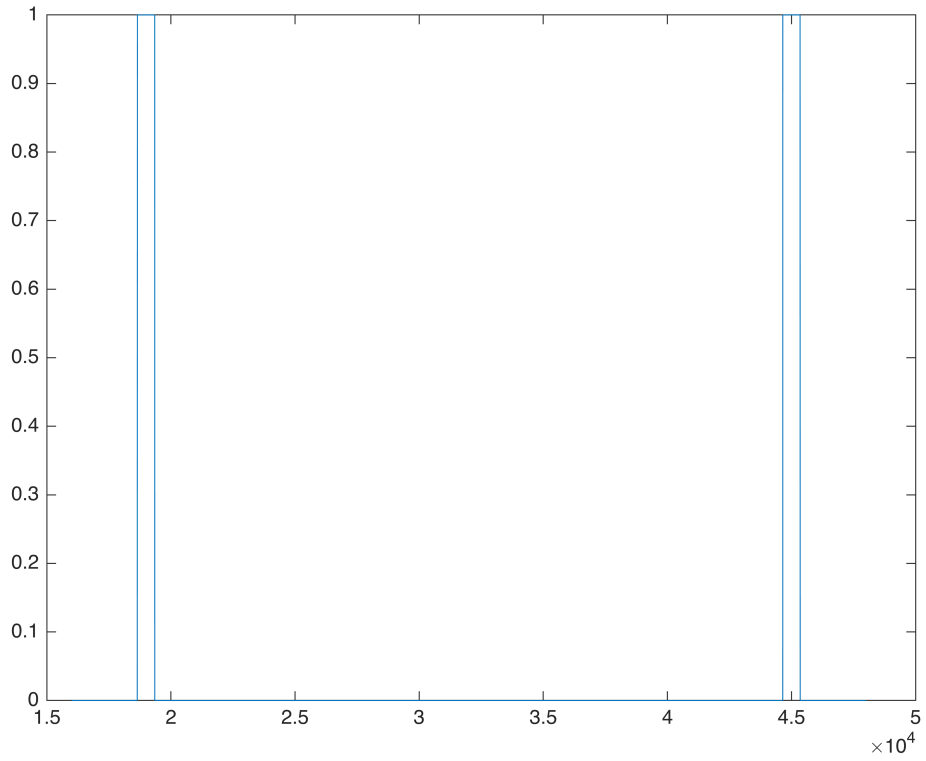




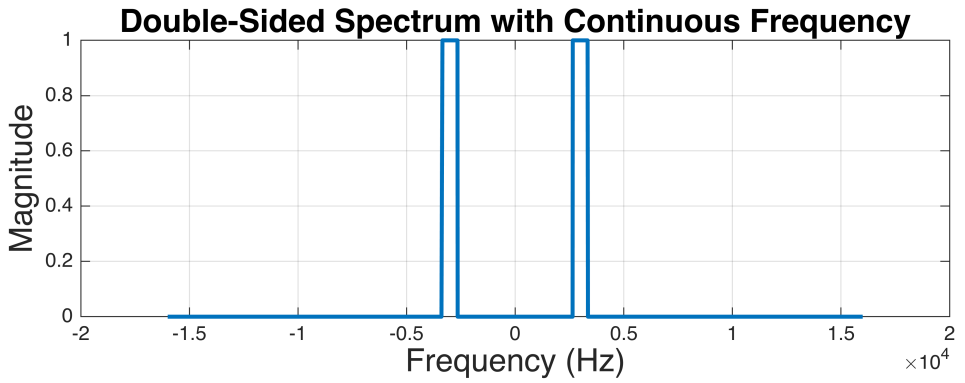
Double-sided spectrum, of the ideal BPF, versus normalized discrete frequency and continuous frequency in a single figure, but as subplots.

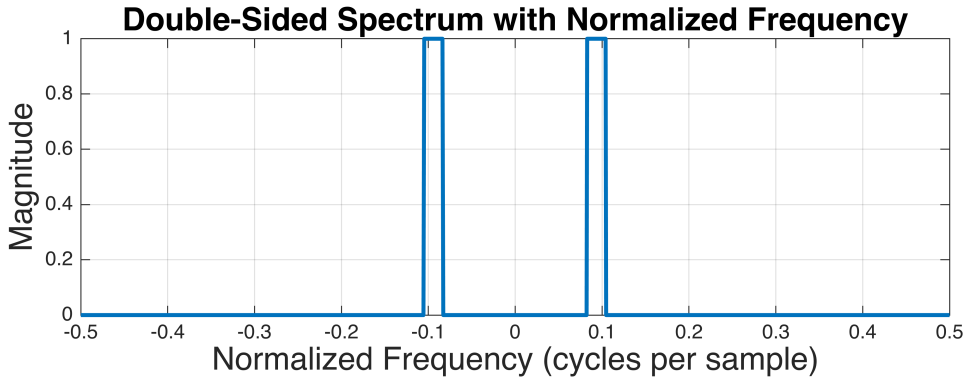
```
clc;
clear all;
close all;

% Define parameters
Fs = 32000;
N1 = 32000;
F = ((0:1:N1-1)/N1)-0.5;
F_range = F*Fs;
fc = 3000; % normalized center frequency
BW = 700/Fs; % normalized bandwidth
% 3700
% 2300

% Create the bandpass filter
BPFilter_Double_Continuous = (rect(F - (fc/Fs), BW/2) + rect(F + (fc/Fs),
BW/2));
BPFilter_Double_Normalized = (rect(F - (fc/Fs), BW/2) + rect(F + (fc/Fs),
BW/2));
BPFilter_Double_Continuous=(BPFilter_Double_Continuous);
```

```
BPFilter_Double_Normalized=(BPFilter_Double_Normalized);
g = fftshift(BPFilter_Double_Continuous);
figure
plot(F_range+(Fs/2), g)
```




```
% Plot the continuous frequency spectrum
figure
subplot(2,1,1)
plot(F_range, BPFilter_Double_Continuous, 'LineWidth', 2);
title('Double-Sided Spectrum with Continuous Frequency', 'FontSize', 16);
xlabel('Frequency (Hz)', 'FontSize', 16);
ylabel('Magnitude', 'FontSize', 16);
grid on;

% Plot the normalized frequency spectrum
subplot(2,1,2)
plot(F, BPFilter_Double_Normalized, 'LineWidth', 2);
title('Double-Sided Spectrum with Normalized Frequency', 'FontSize', 16);
xlabel('Normalized Frequency (cycles per sample)', 'FontSize', 16);
ylabel('Magnitude', 'FontSize', 16);
grid on;
```

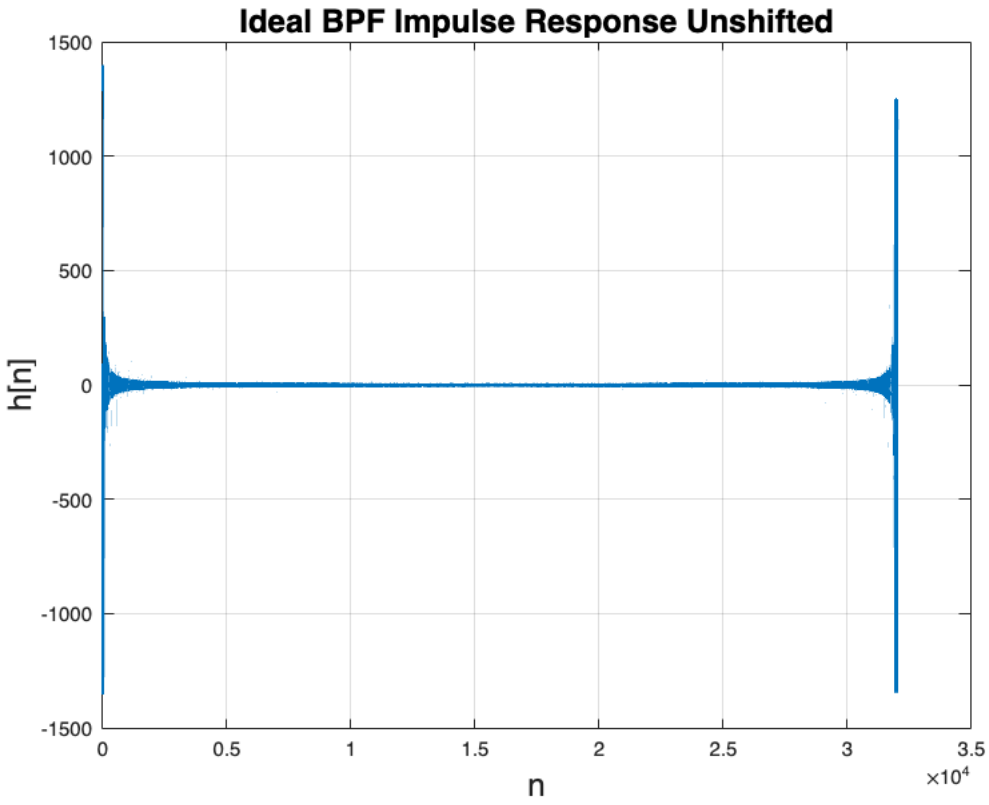




Ideal BPF filter impulse response (unshifted)

```
clc;
clear all;
close all;

% Define parameters
Fs = 32000;
N1 = 32000;
F = ((0:1:Fs-1)/N1)-0.5;
fc = 3000; % normalized center frequency
BW = 700/Fs; % normalized bandwidth
% 3700
% 2300

% Create the bandpass filter
BPFilter = (rect(F - (fc/Fs), BW/2) + rect(F + (fc/Fs), BW/2));

% Compute inverse DFT to get impulse response
[hn_single, n] = IDFT1(fftshift(BPFilter));
hn_double = real(hn_single);

% Plot impulse response
figure;
```

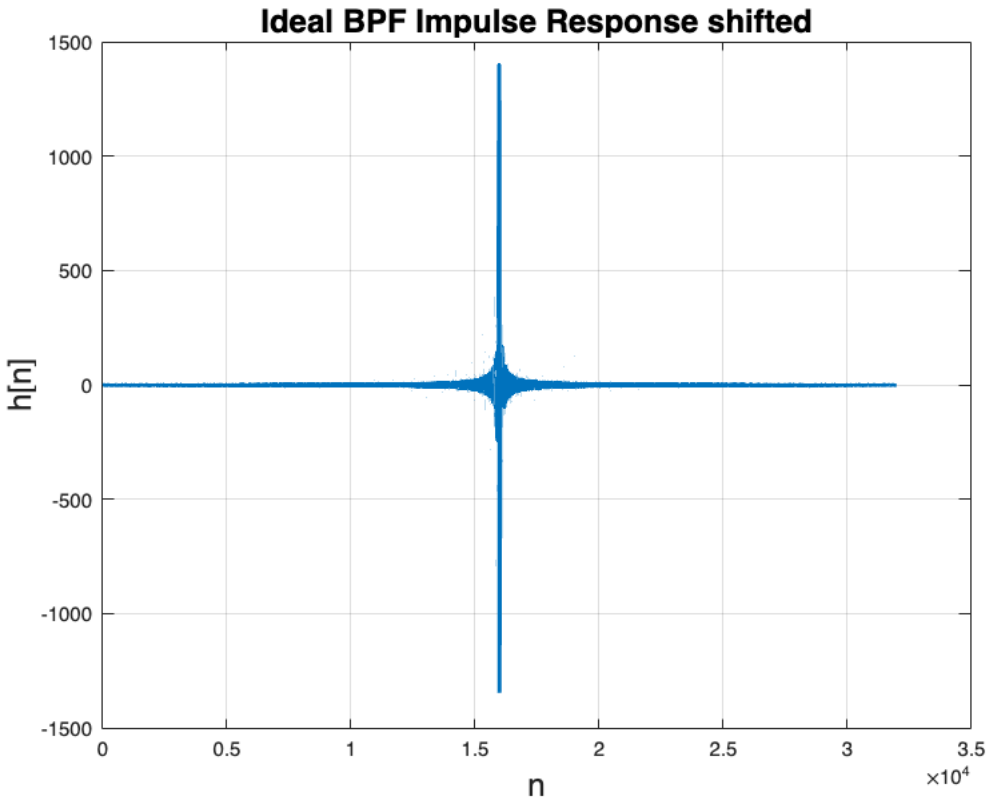
```
plot(n, hn_double, 'LineWidth', 2); % Taking real part because impulse
response should be real
title('Ideal BPF Impulse Response Unshifted', 'FontSize', 16);
xlabel('n', 'FontSize', 16);
ylabel('h[n]', 'FontSize', 16);
grid on;
```



Ideal BPF filter impulse response (shifted)

```
clc;
clear all;
close all;

% Define parameters
Fs = 32000;
N1 = 32000;
F = ((0:1:Fs-1)/N1)-0.5;
fc = 3000; % normalized center frequency
BW = 700/Fs; % normalized bandwidth
% 3700
% 2300

% Create the bandpass filter
BPFilter = (rect(F - (fc/Fs), BW/2) + rect(F + (fc/Fs), BW/2));

% Compute inverse DFT to get impulse response
```

```
[hn_single, n] = IDFT1(fftshift(BPFilter));
hn_single = real(fftshift(hn_single));

% Plot impulse response
figure;
plot(n, hn_single, 'LineWidth', 2); % Taking real part because impulse
response should be real
title('Ideal BPF Impulse Response shifted', 'FontSize', 16);
xlabel('n', 'FontSize', 16);
ylabel('h[n]', 'FontSize', 16);
grid on;
```



Subplots showing the following in a single figure

o The impulse response

o Chosen window function (for N-tap FIR filter)

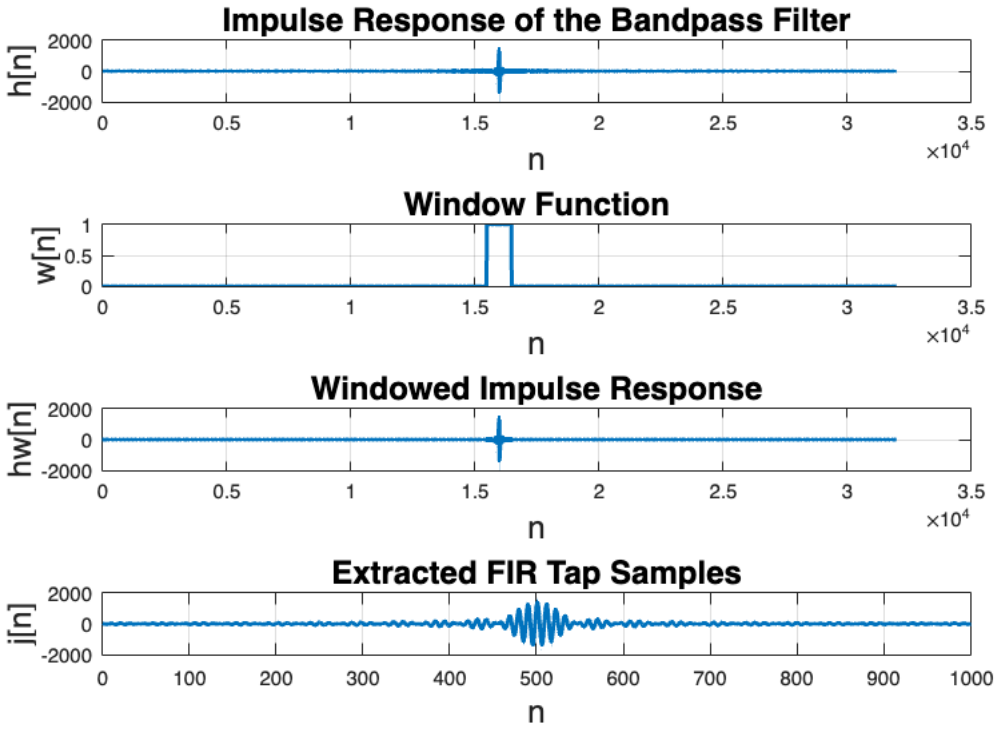
o The response after multiplication with the window function

o Extracted FIR tap samples.

```
clc;
```

```
clear all;
close all;

% Define parameters
Fs = 32000;     % Sampling frequency
N1 = 32000;     % Number of samples
F = ((0:1:Fs-1)/N1)-0.5; % Normalized frequency range
fc = 3000;      % Center frequency
BW = 700/Fs;    % Bandwidth

% Create the bandpass filter
BPFilter = (rect(F - (fc/Fs), BW/2) + rect(F + (fc/Fs), BW/2));

% Compute inverse DFT to get impulse response
[hn_single, n] = IDFT1(fftshift(BPFilter));
hn_single = real(fftshift(hn_single));

% Define window function parameters and window function
taps = 1000;
width = floor(taps/2);
window = rect(n-(length(n)/2), width);

% Apply window function to impulse response
htapped = hn_single .* window;
htappednew = htapped((N1/2)-(width):(N1/2)+(width));

% Plot impulse response
subplot(4,1,1)
plot(n, hn_single, 'LineWidth', 2);
title('Impulse Response of the Bandpass Filter', 'FontSize', 16);
xlabel('n', 'FontSize', 16);
ylabel('h[n]', 'FontSize', 16);
grid on;

% Plot window function
subplot(4,1,2)
plot(n, window, 'LineWidth', 2);
title('Window Function', 'FontSize', 16);
xlabel('n', 'FontSize', 16);
ylabel('w[n]', 'FontSize', 16);
grid on;

% Plot impulse response after window function application
subplot(4,1,3)
plot(n, htapped, 'LineWidth', 2);
title('Windowed Impulse Response', 'FontSize', 16);
xlabel('n', 'FontSize', 16);
ylabel('hw[n]', 'FontSize', 16);
grid on;
% Plot FIR tap samples
```

```
subplot(4,1,4)
plot(0:taps, htappednew, 'LineWidth', 2);
title('Extracted FIR Tap Samples', 'FontSize', 16);
xlabel('n', 'FontSize', 16);
ylabel('j[n]', 'FontSize', 16);
grid on;
```



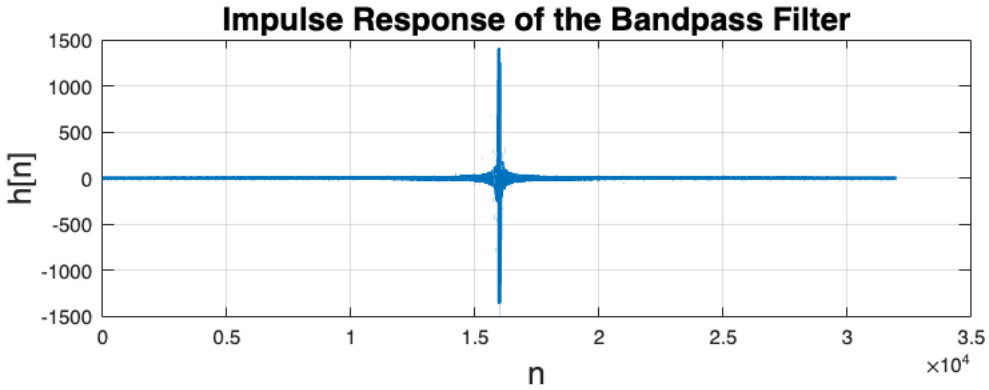
Subplots showing the following plots in a single figure

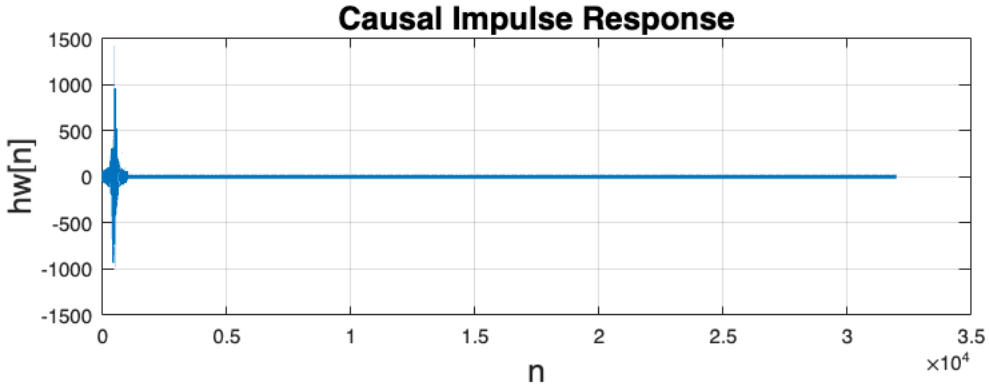
o The impulse response

o The extracted FIR samples with zero padding

```
% Start fresh
clc;
clear all;
close all;

% Define parameters
Fs = 32000;      % Sampling frequency
N1 = 32000;      % Number of samples
F = ((0:1:Fs-1)/N1)-0.5; % Normalized frequency range
fc = 3000;       % Center frequency
```

```
BW = 700/Fs;    % Bandwidth

% Create the bandpass filter
BPFilter = (rect(F - (fc/Fs), BW/2) + rect(F + (fc/Fs), BW/2));

% Compute inverse DFT to get impulse response
[hn_single, n] = IDFT1(fftshift(BPFilter));
hn_single = real(fftshift(hn_single));

% Define window function parameters and window function
taps = 1000;
width = floor(taps/2);
window = rect(n-(length(n)/2), width);

% Apply window function to impulse response
htapped = hn_single .* window;
htappednew = htapped((N1/2)-(width):(N1/2)+(width));
htapped1 = [htappednew zeros(1,length(n)-length(htappednew))];
[Htap, frrr]= DFT1(htapped1);
% Plot impulse response
subplot(2,1,1)
plot(n, hn_single, 'LineWidth', 2);
title('Impulse Response of the Bandpass Filter', 'FontSize', 16);
xlabel('n', 'FontSize', 16);
ylabel('h[n]', 'FontSize', 16);
grid on;

% Plot impulse response after window function application
subplot(2,1,2)
plot(n, htapped1, 'LineWidth', 2);
title('Causal Extracted Impulse Response', 'FontSize', 16);
xlabel('n', 'FontSize', 16);
ylabel('hw[n]', 'FontSize', 16);
grid on;
```

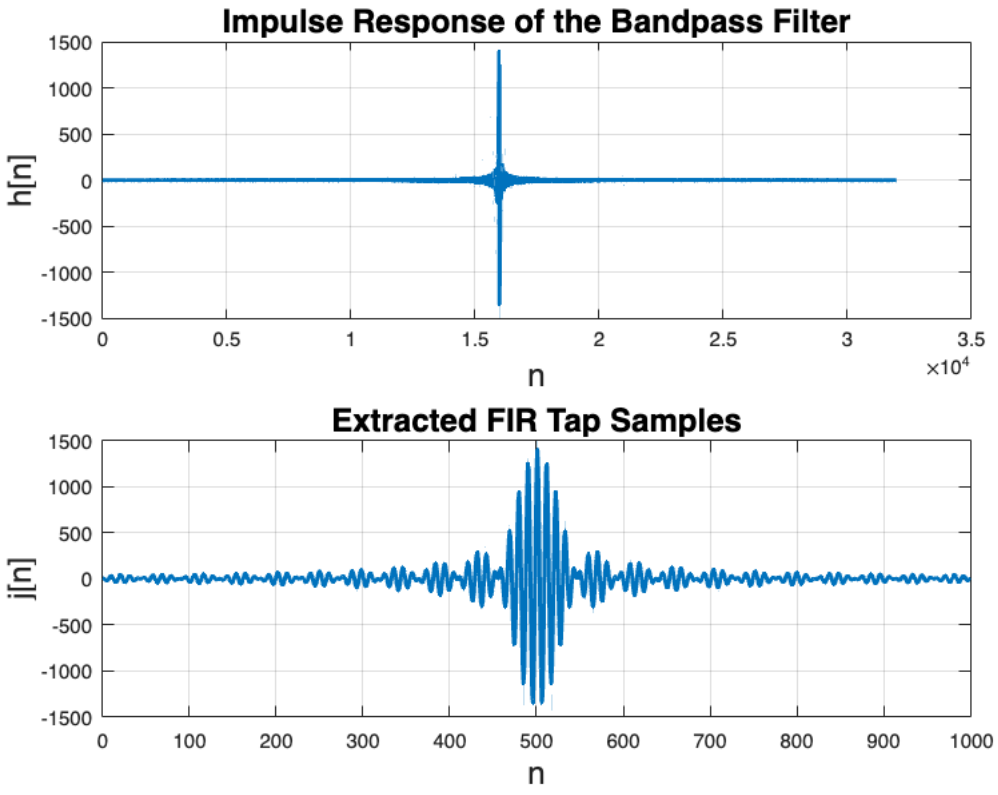




A plot comparing the ideal frequency response and the frequency response of the extracted impulse response samples. (You may use the plot function to improve visual comparison).

```
clc;
clear all;
close all;

% Define parameters
Fs = 32000;     % Sampling frequency
N1 = 32000;     % Number of samples
F = ((0:1:Fs-1)/N1)-0.5; % Normalized frequency range
fc = 3000;      % Center frequency
BW = 700/Fs;    % Bandwidth

% Create the bandpass filter
BPFilter = (rect(F - (fc/Fs), BW/2) + rect(F + (fc/Fs), BW/2));

% Compute inverse DFT to get impulse response
[hn_single, n] = IDFT1(fftshift(BPFilter));
hn_single = real(fftshift(hn_single));
```

```
% Define window function parameters and window function
taps = 1000;
width = floor(taps/2);
window = rect(n-(length(n)/2), width);

% Apply window function to impulse response
htapped = hn_single .* window;
htappednew = htapped((N1/2)-(width):(N1/2)+(width));

% Plot impulse response
subplot(2,1,1)
plot(n, hn_single, 'LineWidth', 2);
title('Impulse Response of the Bandpass Filter', 'FontSize', 16);
xlabel('n', 'FontSize', 16);
ylabel('h[n]', 'FontSize', 16);
grid on;

% Plot FIR tap samples
subplot(2,1,2)
plot(0:taps, htappednew, 'LineWidth', 2);
title('Extracted FIR Tap Samples', 'FontSize', 16);
xlabel('n', 'FontSize', 16);
ylabel('j[n]', 'FontSize', 16);
grid on;
```



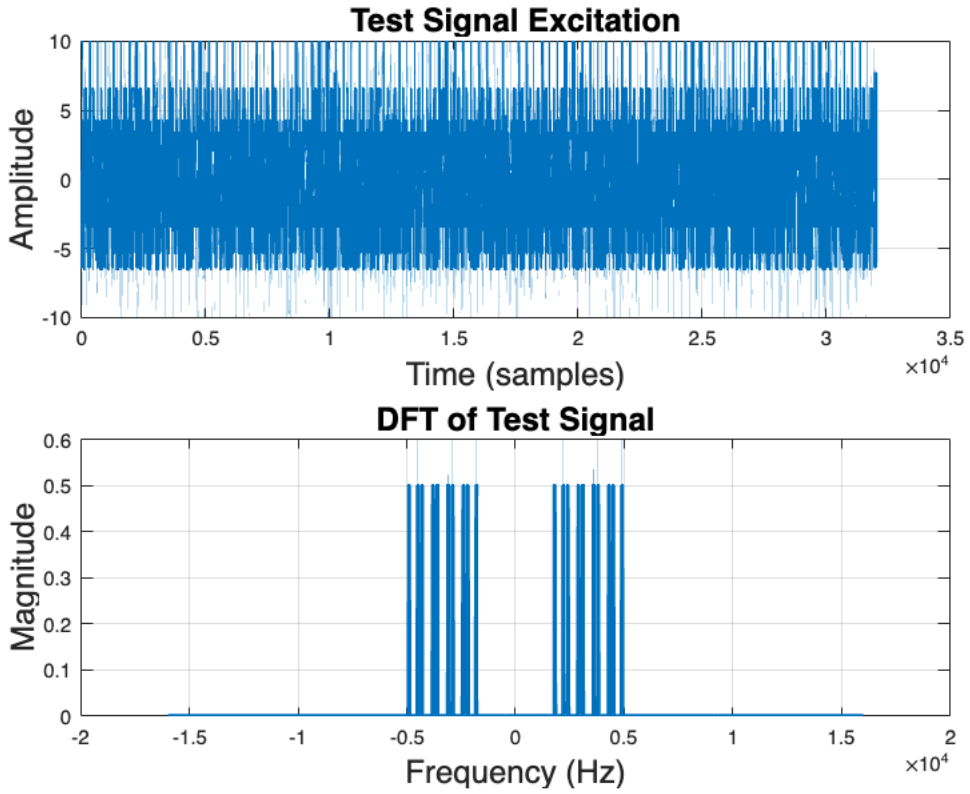
A plot showing the test signal excitation and its DFT output

```
clc;
clear all;
close all;

% Define parameters
Fs = 32000;
N1 = 32000;
F = ((0:1:Fs-1)/N1)-0.5;
fc = 3000; % normalized center frequency
BW = 700/Fs; % normalized bandwidth
% 3700
% 2300

% Generate signal x
N0 = length(N1);
n_excitation = 0:N1-1;
% Generate signal x
% The signal is a sum of cosines with different frequencies that are well
within the lower stop band,
% within the lower transition band, within the pass band, within the upper
transition band,
% and well within the upper stop band of the ideal frequency response.
% Each cosine has a frequency specified by the multiplier of the time
variable
% in the argument of the cosine function.
```

The frequencies are: 1800, 2200, 2400, 2900, 3100, 3600, 3800, 4300, 4500, and 4900.

```
x = cos(2*pi*1800*(n_excitation/N1)) + cos(2*pi*2200*(n_excitation/N1)) +
cos(2*pi*2400*(n_excitation/N1)) + cos(2*pi*2900*(n_excitation/N1))...
    + cos(2*pi*3100*(n_excitation/N1)) + cos(2*pi*3600*(n_excitation/N1)) +
cos(2*pi*3800*(n_excitation/N1)) + cos(2*pi*4300*(n_excitation/N1))...
    + cos(2*pi*4500*(n_excitation/N1)) + cos(2*pi*4900*(n_excitation/N1));
[xdft, fnew] = DFT1(x);

% Plot the test signal
figure;
subplot(2,1,1);
plot(n_excitation, x, 'LineWidth', 2);
title('Test Signal Excitation', 'FontSize', 16);
xlabel('Time (samples)', 'FontSize', 16);
ylabel('Amplitude', 'FontSize', 16);
grid on;

% Plot the DFT of the test signal
```

```
subplot(2,1,2);
plot(fnew*Fs, fftshift(abs(xdft)), 'LineWidth', 2);
title('DFT of Test Signal', 'FontSize', 16);
xlabel('Frequency (Hz)', 'FontSize', 16);
ylabel('Magnitude', 'FontSize', 16);
grid on;
```



A plot showing the filter response using convolution and its spectral profile.

o Show a comparison spectrum comparing the before and after filtering

```
clc;
clear all;
close all;

% Define parameters
Fs = 32000;
N1 = 32000;
F = ((0:1:Fs-1)/N1)-0.5;
fc = 3000; % normalized center frequency
BW = 700/Fs; % normalized bandwidth
% 3700
```

```
% 2300

% Create the bandpass filter
BPFilter = (rect(F - (fc/Fs), BW/2) + rect(F + (fc/Fs), BW/2));

% Compute inverse DFT to get impulse response
[hn_single, n] = IDFT1(fftshift(BPFilter));
hn_single = real(fftshift(hn_single));
n=0:1:N1-1;
taps = 1000;
width = floor(taps/2);
window = rect(n-(length(n)/2), width);
htapped = hn_single .* window;
% figure
% plot(n, htapped)
htappednew = htapped((N1/2)-(width):(N1/2)+(width));

% Generate signal x
n_excitation = 0:N1-1;
% Generate signal x
% The signal is a sum of cosines with different frequencies that are well
within the lower stop band,
% within the lower transition band, within the pass band, within the upper
transition band,
% and well within the upper stop band of the ideal frequency response.
% Each cosine has a frequency specified by the multiplier of the time
variable
% in the argument of the cosine function.

% The frequencies are: 1800, 2200, 2400, 2900, 3100, 3600, 3800, 4300,
4500, and 4900.
```

Passing Frequencies should be 2900 & 3100

```
x = cos(2*pi*1800*(n_excitation/N1)) + cos(2*pi*2200*(n_excitation/N1)) +
cos(2*pi*2400*(n_excitation/N1)) + cos(2*pi*2900*(n_excitation/N1))...
    + cos(2*pi*3100*(n_excitation/N1)) + cos(2*pi*3600*(n_excitation/N1)) +
cos(2*pi*3800*(n_excitation/N1)) + cos(2*pi*4300*(n_excitation/N1))...
    + cos(2*pi*4500*(n_excitation/N1)) + cos(2*pi*4900*(n_excitation/N1));
%convolution and DFT of signals
tic;
y_conv = conv(x, htappednew, 'same');
```

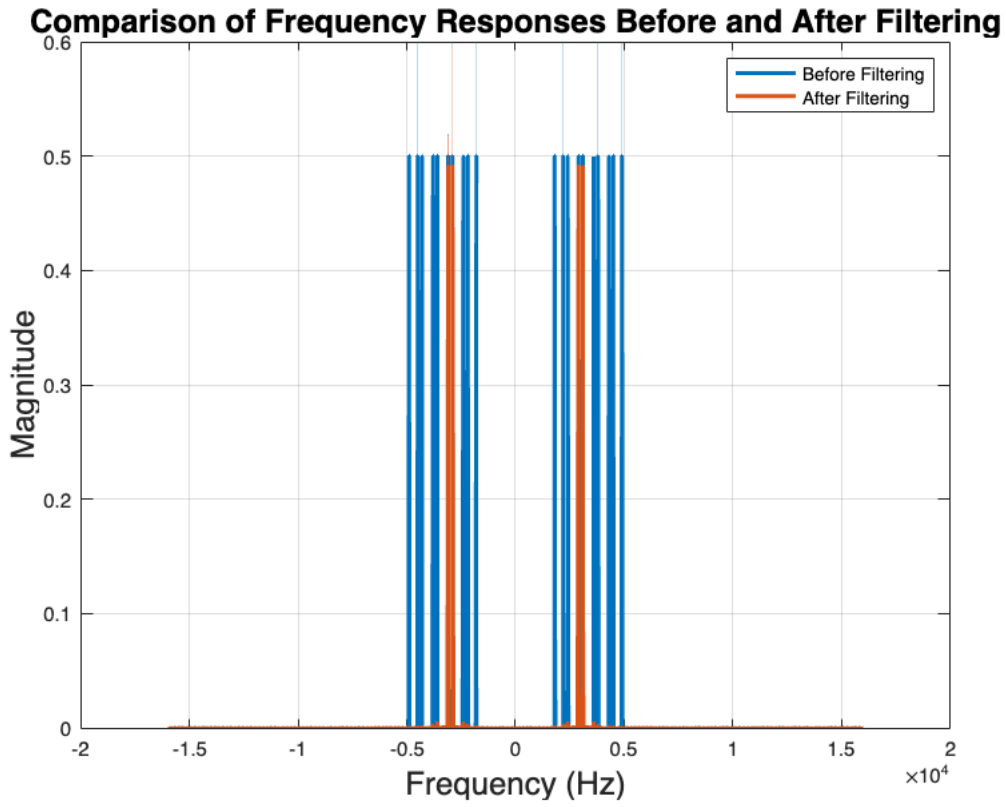
```
ans = 32000
```

```
toc;
```

```
Elapsed time is 0.061443 seconds.
```

```
[yf, nnew] = DFT1(y_conv);
% Calculate the DFT of the original signal x
```

```
[xdft, fnew] = DFT1(x);

% Create a new figure for the comparison
figure;

% Plot the DFT of the original signal
plot(nnew*Fs, fftshift(abs(xdft)), 'LineWidth', 2);
hold on; % This allows us to plot the next line on the same figure

% Plot the DFT of the convolved signal
plot(nnew*Fs, fftshift(abs(yf))/Fs, 'LineWidth', 2);

% Add title and labels
title('Comparison of Frequency Responses Before and After Filtering',
'FontSize', 16);
xlabel('Frequency (Hz)', 'FontSize', 16);
ylabel('Magnitude', 'FontSize', 16);
legend('Before Filtering', 'After Filtering'); % Add a legend to
distinguish the lines
grid on;
```



A plot showing the filter response obtained using the difference equation and its spectral profile. o Show a comparison spectrum comparing the before

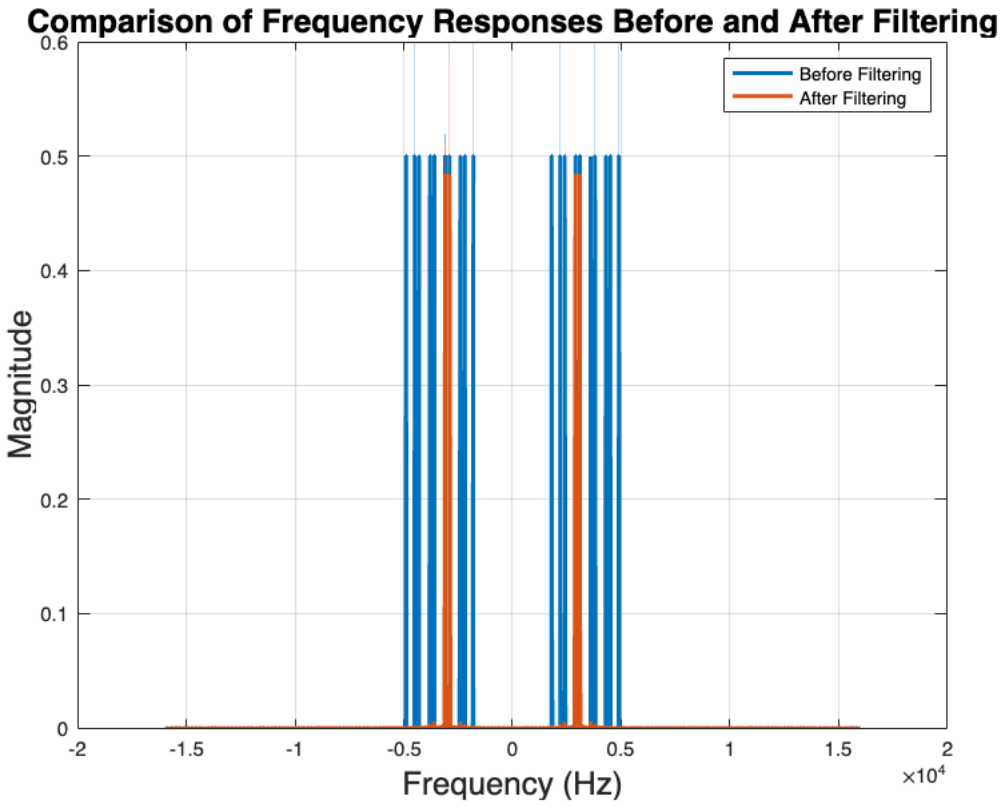
and after filtering frequency responses in the same plot area but different colors.

```
clc;
clear all;
close all;

% Define parameters
Fs = 32000;
N1 = 32000;
F = ((0:1:Fs-1)/N1)-0.5;
fc = 3000; % normalized center frequency
BW = 700/Fs; % normalized bandwidth
% 3700
% 2300

% Create the bandpass filter
BPFilter = (rect(F - (fc/Fs), BW/2) + rect(F + (fc/Fs), BW/2));

% Compute inverse DFT to get impulse response
[hn_single, n] = IDFT1(fftshift(BPFilter));
hn_single = real(fftshift(hn_single));
n=0:1:N1-1;
taps = 1000;
width = floor(taps/2);
window = rect(n-(length(n)/2), width);
htapped = hn_single .* window;
% figure
% plot(n, htapped)
htappednew = htapped((N1/2)-(width):(N1/2)+(width));

% Generate signal x
n_excitation = 0:N1-1;
x = cos(2*pi*1800*(n_excitation/N1)) + cos(2*pi*2200*(n_excitation/N1)) +
cos(2*pi*2400*(n_excitation/N1)) + cos(2*pi*2900*(n_excitation/N1))...
    + cos(2*pi*3100*(n_excitation/N1)) + cos(2*pi*3600*(n_excitation/N1)) +
cos(2*pi*3800*(n_excitation/N1)) + cos(2*pi*4300*(n_excitation/N1))...
    + cos(2*pi*4500*(n_excitation/N1)) + cos(2*pi*4900*(n_excitation/N1));

H = htappednew;

% Length of the filter
N = length(H);

% Preallocate the output signal
y1 = zeros(1, length(x));


tic;
```

```
for i = 1:length(x)
    for j = 1:N
        if (i-j+1)>0
            y1(i) = y1(i) + H(j) * x(i-j+1);
        end
    end
end
toc;
```

```
Elapsed time is 0.267096 seconds.
```

```
[yf, nnew] = DFT1(y1);

% Calculate the DFT of the original signal x
[xdft, fnew] = DFT1(x);

% Create a new figure for the comparison
figure;

% Plot the DFT of the original signal
plot(fnew*Fs, fftshift(abs(xdft)), 'LineWidth', 2);
hold on; % This allows us to plot the next line on the same figure

% Plot the DFT of the convolved signal
plot(nnew*Fs, fftshift(abs(yf))/Fs, 'LineWidth', 2);

% Add title and labels
title('Comparison of Frequency Responses Before and After Filtering',
'FontSize', 16);
xlabel('Frequency (Hz)', 'FontSize', 16);
ylabel('Magnitude', 'FontSize', 16);
legend('Before Filtering', 'After Filtering'); % Add a legend to
distinguish the lines
grid on;
```



The analysis of how the filter response changes as N changes.

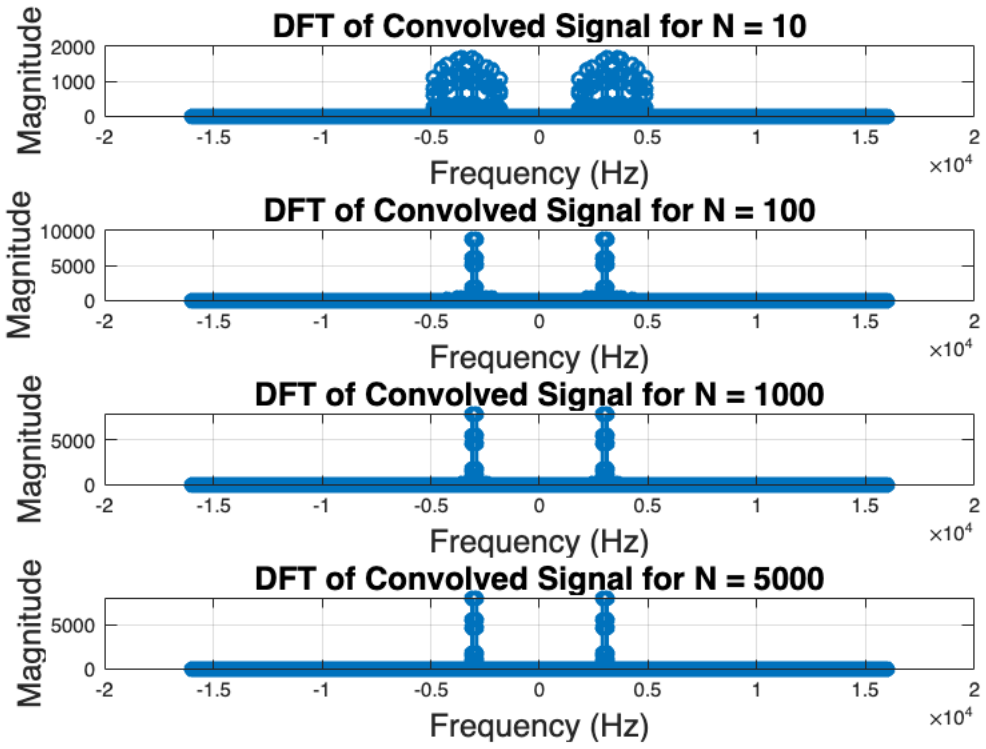
o Your analysis must show the effect of at least 4 different creatively chosen values of N. Discuss the effect on filter performance as N increases and decreases.

o Choose four significantly different values of N so that you have good results for your discussion section.

```
% Define parameters
Fs = 32000;
N1 = 32000;
F = ((0:1:Fs-1)/N1)-0.5;
fc = 3000; % normalized center frequency
BW = 700/Fs; % normalized bandwidth

% Create the bandpass filter
BPFilter = (rect(F - (fc/Fs), BW/2) + rect(F + (fc/Fs), BW/2));
```

```
% Compute inverse DFT to get impulse response
[hn_single, n] = IDFT1(fftshift(BPFilter));
hn_single = real(fftshift(hn_single));
n=0:1:N1-1;

% Define taps
taps_values = [10, 50, 1000, 5000]; % different values of N (taps)

% Generate signal x
n_excitation = 0:N1-1;

% The signal is a sum of cosines with different frequencies that are well
within the lower stop band,
% within the lower transition band, within the pass band, within the upper
transition band,
% and well within the upper stop band of the ideal frequency response.
% Each cosine has a frequency specified by the multiplier of the time
variable
% in the argument of the cosine function.
x = cos(2*pi*1800*(n_excitation/N1)) + cos(2*pi*2200*(n_excitation/N1)) +
cos(2*pi*2400*(n_excitation/N1)) + cos(2*pi*2900*(n_excitation/N1))...
    + cos(2*pi*3100*(n_excitation/N1)) + cos(2*pi*3600*(n_excitation/N1)) +
cos(2*pi*3800*(n_excitation/N1)) + cos(2*pi*4300*(n_excitation/N1))...
    + cos(2*pi*4500*(n_excitation/N1)) + cos(2*pi*4900*(n_excitation/N1));

% Create a figure
figure;

% Iterate over taps values
for i = 1:length(taps_values)
    taps = taps_values(i);
    width = floor(taps/2);
    window = rect(n-(length(n)/2), width);
    htapped = hn_single .* window;
    htappednew = htapped((N1/2)-(width):(N1/2)+(width));
    htapped = [htappednew zeros(1,length(n)-length(htappednew))];

    %convolution and DFT of signals
    y_conv = conv(x, htappednew);
    [yf, nnew] = DFT1(y_conv);

    % Plot convolved signal
    subplot(length(taps_values),1,i)
    stem(nnew*Fs, fftshift(abs(yf)), 'LineWidth', 2);
    title(['DFT of Convolved Signal for N = ', num2str(taps)], 'FontSize',
16);
    xlabel('Frequency (Hz)', 'FontSize', 16);
    ylabel('Magnitude', 'FontSize', 16);
    grid on;
```

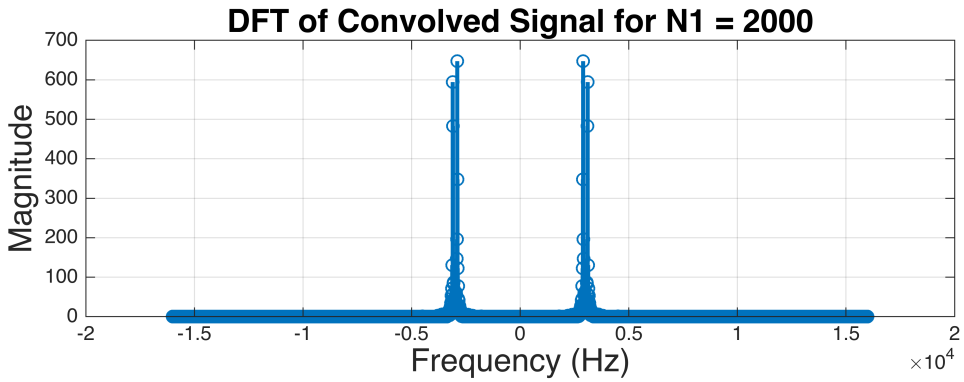
```
end
```

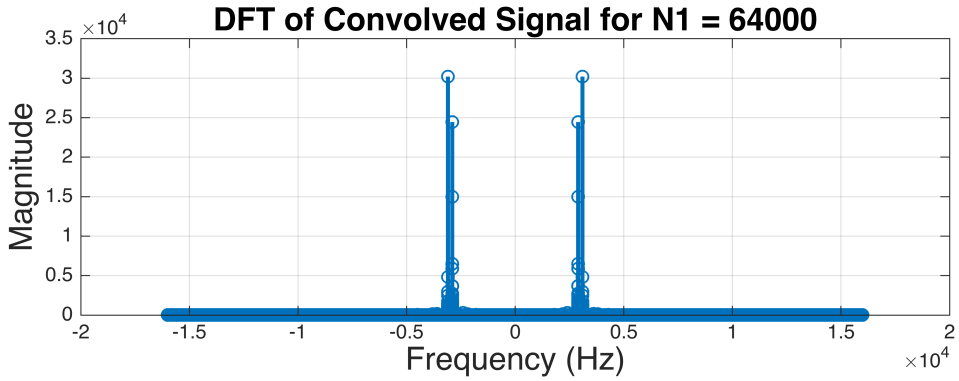


The analysis of how the filter response changes as N1 changes. o Your analysis must show the effect of at least 2 different creatively chosen values of N1 (keep N constant). Is there any change in the filter performance as N1 changes? o Choose two significantly different values of N1 so that you have good results for your discussion section.

```
% Define parameters
Fs = 32000;
N1_values = [2000, 2*32000]; % two different values of N1
fc = 3000; % normalized center frequency
BW = 700/Fs; % normalized bandwidth
taps = 1000;

% Create a figure
figure;

% Iterate over N1 values
for i = 1:length(N1_values)
```

```
    N1 = N1_values(i);
    F = ((0:1:N1-1)/N1)-0.5;

    % Create the bandpass filter
    BPFilter = (rect(F - (fc/Fs), BW/2) + rect(F + (fc/Fs), BW/2));

    % Compute inverse DFT to get impulse response
    [hn_single, n] = IDFT1(fftshift(BPFilter));
    hn_single = real(fftshift(hn_single));
    n=0:1:N1-1;
    width = floor(taps/2);
    window = rect(n-(length(n)/2), width);
    htapped = hn_single .* window;
    htappednew = htapped((N1/2)-(width):(N1/2)+(width));

    % Generate signal x
    n_excitation = 0:N1-1;
    x = cos(2*pi*1800*(n_excitation/Fs)) + cos(2*pi*2200*(n_excitation/Fs))
+ cos(2*pi*2400*(n_excitation/Fs)) + cos(2*pi*2900*(n_excitation/Fs))...
        + cos(2*pi*3100*(n_excitation/Fs)) + cos(2*pi*3600*(n_excitation/
Fs)) + cos(2*pi*3800*(n_excitation/Fs)) + cos(2*pi*4300*(n_excitation/
Fs))...
        + cos(2*pi*4500*(n_excitation/Fs)) + cos(2*pi*4900*(n_excitation/
Fs));

    %convolution and DFT of signals
    y_conv = conv(x, htappednew);
    [yf, nnew] = DFT1(y_conv);

    % Plot convolved signal
    subplot(length(N1_values),1,i)
    stem(nnew*Fs, fftshift(abs(yf)), 'LineWidth', 2);
    title(['DFT of Convolved Signal for N1 = ', num2str(N1)], 'FontSize',
16);
    xlabel('Frequency (Hz)', 'FontSize', 16);
    ylabel('Magnitude', 'FontSize', 16);
    grid on;
end
```





The implementation of the Finite Impulse Response (FIR) filter in this context was initially designed for a fixed number of extracted samples, derived from the obtained filter impulse response. This number then becomes the quantity of taps in the N-tap FIR filter. In digital signal processing, this directly correlates to the filter's order, which significantly affects the filter's performance and characteristics.

In the analysis of how the filter response changes as the number of taps (N) changes, it was observed that lower values of N allowed for more frequencies to pass through. This is mainly because a lower N means fewer coefficients in the filter, resulting in a less selective filter with a broader passband and transition band. Consequently, frequencies that might have been attenuated by a higher order filter may pass through a lower order filter.

On the other hand, as N increased, the filter became more selective, and only frequencies at around 2900 Hz and 3100 Hz were allowed to pass. This effect is due to the higher order of the filter. The more taps a filter has, the more refined its frequency response becomes. This allows it to better attenuate frequencies outside the desired passband. However, it's important to note that a higher order also means more computational complexity and a longer processing time, which can be a significant consideration in real-time applications.

In terms of N1, which is the length of the filter's impulse response, the analysis showed that lower values resulted in lower magnitudes in the frequency response, while higher values of N1 resulted in higher magnitudes. This phenomenon can be related to the conservation of energy principle. As N1 increases, the energy of the filter (represented by the magnitude response) becomes more concentrated in certain frequencies, leading to a higher magnitude response.

However, increasing N1 does not necessarily improve the filter's performance. It may lead to a phenomenon known as time aliasing if not properly managed. This is because as we increase the length of the impulse response, we increase the filter's memory, i.e., the number of past inputs it takes into account. This can cause interference between distant samples, leading to unwanted effects in the filtered signal.

In conclusion, the number of taps N and the length of impulse response N1 play critical roles in shaping the characteristics of a FIR filter. By manipulating these parameters, we can design a filter with specific desired properties, taking into account trade-offs such as filter selectivity, computational complexity, and potential for time aliasing. It's important to consider these factors when designing filters for different applications in digital signal processing.