SUSTech DSP Lab7 Report

Cao Zhengyang
Department of Electronics and Electrical Engineering
12110623@mail.sustech.edu.cn

I. Introduction

This report explores digital filter design, focusing on Finite Impulse Response (FIR) and Infinite Impulse Response (IIR) filters. It investigates the influence of key parameters in FIR and IIR filters and applies windowing techniques for filter design offering practical insights into the trade-offs inherent in designing effective digital filters.

II. Design of a Simple FIR Filter

A. magnitude response of filter

Given the analytical transfer function, we derive the difference function using the following definition:

$$H_f(z) = 1 - 2\cos\theta z^{-1} + z^{-2} = \frac{Y(z)}{X(z)} = \frac{1 - 2\cos\theta z^{-1} + z^{-2}}{1} \tag{1}$$

By incorporating both $X(z)$ in the numerator and denominator through multiplexing, we obtain the difference equation:

$$y[n] = x[n] - 2\cos\theta x[n-1] + x[n-2] \tag{2}$$

The system diagram is as follows:

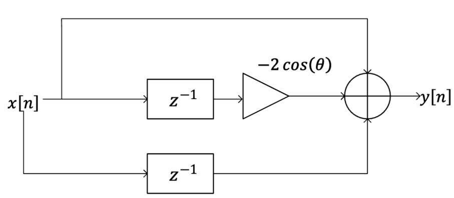

Fig. 1. System Diagram of the Filter

Taking not just three but six values of θ, we illustrate the results in the figure below:

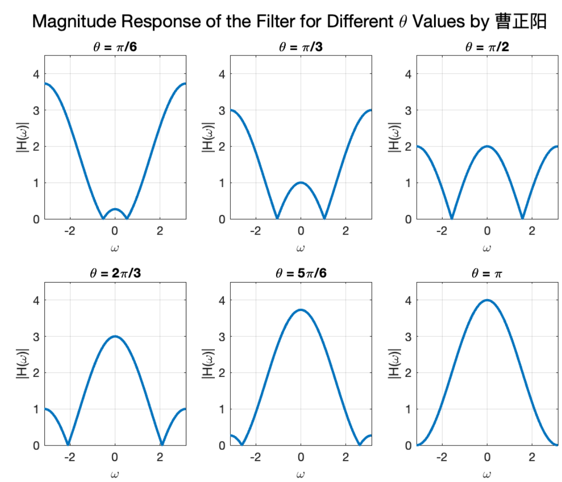

Fig. 2. Magnitude Response Variations for Different θ Values

As θ approaches 0, $|H(0)|$ tends to 0, indicating a low-pass filter behavior. Conversely, as θ approaches π, $|H(0)|$ tends to 4, resembling high-pass filter characteristics. The filter response amplifies near zero and attenuates at $\pm\pi$ as θ varies from 0 to π.

B. Filtering applied to speech

The results are depicted in Figure 3, showcasing the analysis of audio signal and FIR filtering.

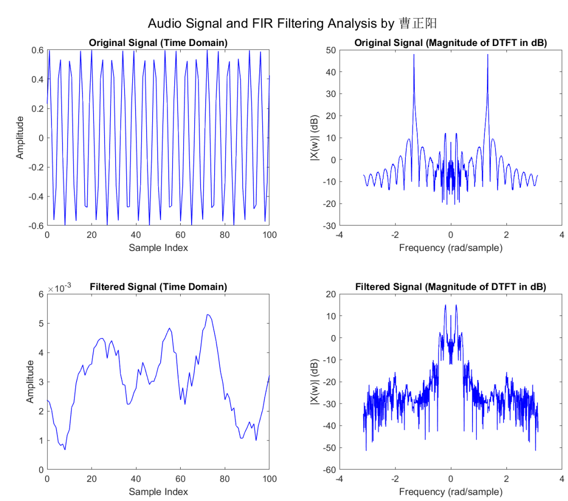

Fig. 3. Audio Signal and FIR Filtering Analysis

The corresponding function is detailed below:

```
function y = FIRfilter(x)
    % Compute DTFT
    [X, w] = DTFT(x, 0);
    % Find the index of the maximum magnitude in the DTFT
    [~, Imax] = max(abs(X));
    % Design the FIR filter
    theta = w(Imax);
    h = [1 -2*cos(theta) 1];
    % Apply the FIR filter using convolution
    y = conv(x, h, 'same');
end
```

As the given speech saying: ”please get rid of this beep”, the original speech contains an audible beep. After applying the designed FIR filter, the beep is effectively filtered out. Examination of the two DTFT plots reveals distinct peaks around 50dB corresponding to the beeping sound. Post-filtering, these peaks disappeared. Audibly, the filtered speech confirms the elimination of the beep.

And by the way, this filter is a band-pass filter.

III. Design of a Simple IIR Filter

A. Magnitude response of filter

Given the analytical transfer function, we derive the difference function using the following definition:

$$H_i(z) = \frac{1-r}{1-2r\cos\theta z^{-1} + r^2 z^{-2}} = \frac{Y(z)}{X(z)} \tag{3}$$

By employing the analytical transfer function $H_i(z)$ and applying the definition $H_i(z) = \frac{Y(z)}{X(z)}$, we can derive the corresponding difference equation:

$$y[n] - 2r\cos\theta y[n-1] + r^2 y[n-2] = (1-r)x[n] \tag{4}$$

The system diagram is as follows:

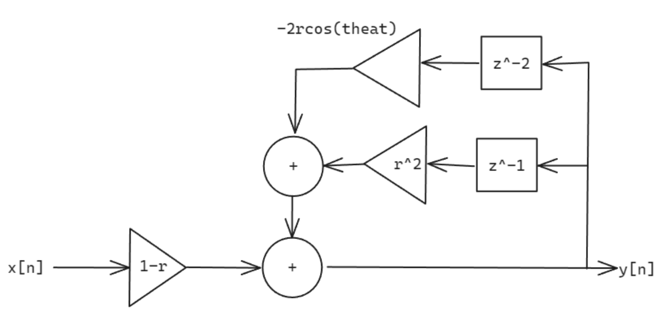

Fig. 4. System Diagram of the Filter

Taking not just three but five values of r, we illustrate the results in the figure below:

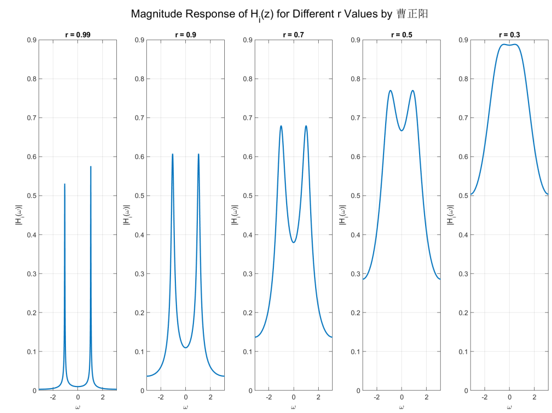

Fig. 5. Magnitude Response Variations for Different r Values

As r increases, we can observe that:

- Bandwidth (width of the peak) decreases, resulting in a more selective filter.
- Peak magnitude around θ increases.

The reason for this is because larger values of r bring the poles closer to the unit circle, which can compromise stability. As poles approach the unit circle, the denominator of the transfer function approaches zero more rapidly as frequency approaches the pole angle. This, in turn, leads to a sharper peak in the magnitude response, indicating a narrower bandwidth.

B. Filtering applied to speech

The results are depicted in Figure **??**, showcasing the analysis of audio signal and IIR filtering.

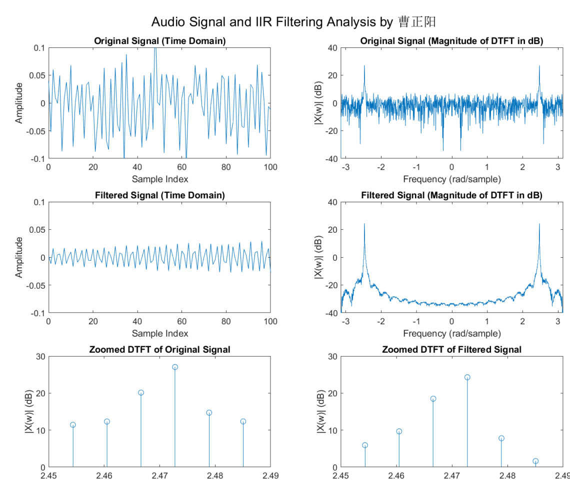

Fig. 6. Audio Signal and FIR Filtering Analysis

The corresponding function is detailed below:

```
function y = IIRfilter(x)
    % Parameters
    theta = (3146 / 8000) * 2 * pi;
```

```
    r = 0.995;
    % Length of input signal
    N = length(x);
    % Initialize output signal
    y = zeros(1, N);
    % Apply recursive difference equation
    y(1) = (1-r)*x(1);
    y(2) = (1-r)*x(2) + 2 * r * cos(theta
        )*y(1);
    for i = 3:N
        y(i) = (1-r)*x(i) + 2 * r * cos(
            theta) * y(i - 1) - (r^2) * y(
            i - 2);
    end
end
```

Similar to the FIR filter applied to the speech, the background noise has been successfully eliminated; however, this has resulted in a reduction in overall volume.

When the value of r is changed to 0.9999999, the sound becomes significantly faint, requiring the volume to be set very high to perceive the audio. The plot is as follows:

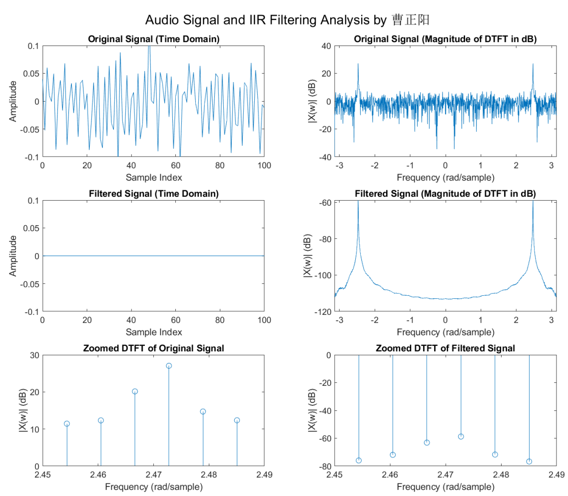

Fig. 7. Audio Signal and FIR Filtering Analysis

We can find that the magnitude response is as low as $-60dB$, that why we can't hear anything. That is why such a value of r is not recommended.

IV. Filter Design Using Truncation

The frequency response of the two filters is illustrated below, highlighting distinct regions as rectangles to signify the passband, transition band, and stopband:

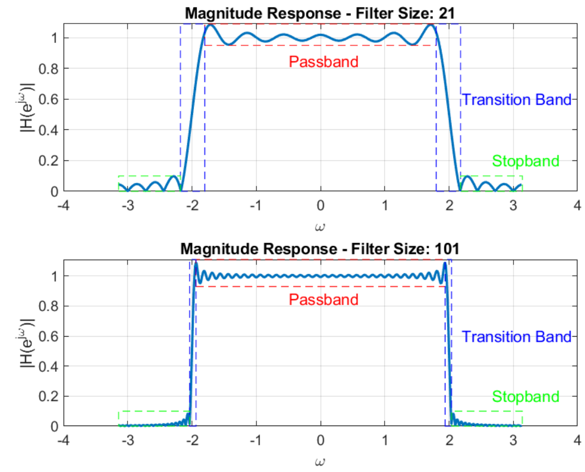

Fig. 8. Magnitude Response of FIR Filters - Passband, Transition Band, and Stopband

The same frequency response is presented in decibels (dB):

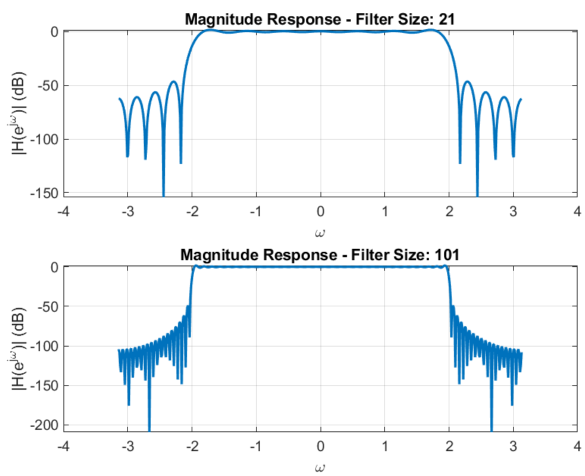

Fig. 9. Magnitude Response of FIR Filters in Decibels

In filters designed by truncating an ideal response, larger filter sizes (N) lead to **reduced stopband ripple** because they better approximate the ideal filter's smooth transition and suppress the ripple-causing sidelobes of the window function. However, this comes at the cost of increased computational complexity and a potentially sharper transition band, necessitating careful consideration of performance and design tradeoffs.

As for the quality of the filtered signals, larger filter sizes generally result in better audio quality by reducing unwanted signal components. However, using larger filters can increase computational complexity. Therefore, the choice of filter size should consider a balance between improved signal separation and computational efficiency based on specific needs.

V. Filter Design Using Standard Windows

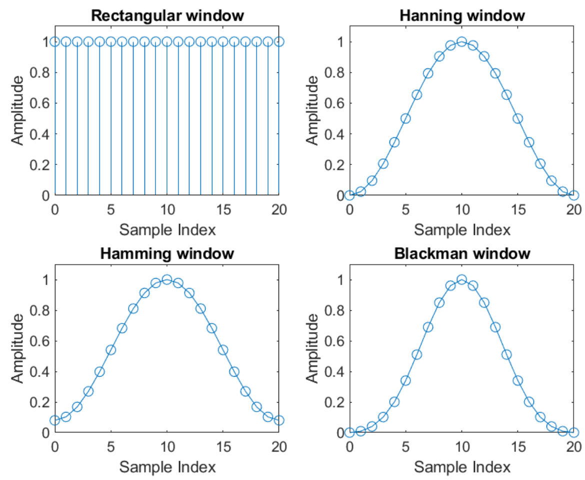

Fig. 10. Time domain plots of the four window functions.

Figure 10 shows their time domain behavior, revealing amplitude variations across sample indices.

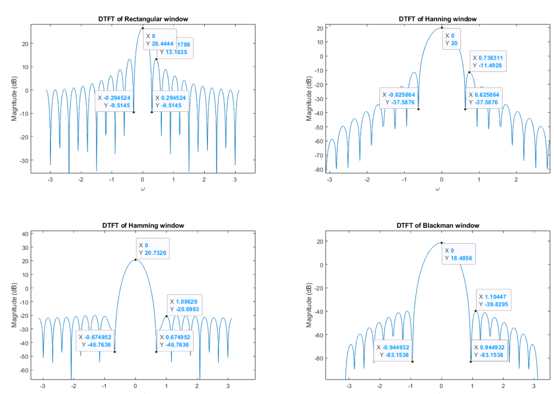

Fig. 11. Frequency domain representation of the four window functions.

I measured the spectrum parameters of each window function using coordinates and compared them with the theoretical values. Table I presents the measured mainlobe width (ω) and peak-to-sidelobe amplitude (in decibels) for the Rectangular, Hanning, Hamming, and Blackman windows.

Window Type	Mainlobe Width (ω)	Peak-to-sidelobe Amp (dB)
Rectangular	0.589	13.2809
Hanning	1.2516	31.4928
Hamming	1.3498	41.4340
Blackman	1.8898	58.3130

TABLE I
MEASURED SPECTRUM PARAMETERS OF WINDOW FUNCTIONS.

In comparing the experimental data with theoretical values, we observe a close agreement between the two datasets.

Furthermore, an interesting trend emerges: as the main lobe width increases, the Peak-to-sidelobe Amplitude also tends to rise. This relationship suggests a noteworthy correlation between the width of the main lobe and the amplitude of sidelobes.

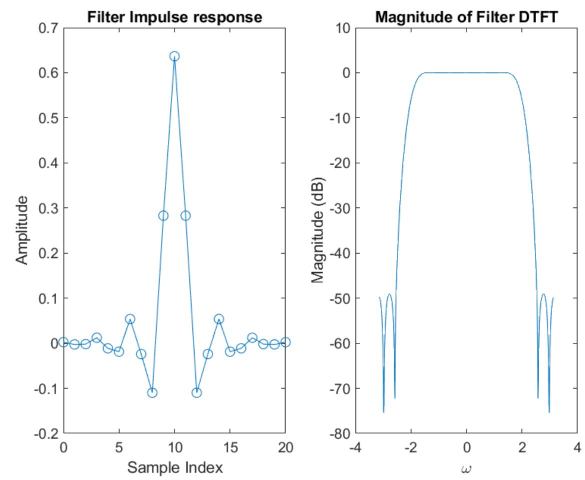

Fig. 12. Impulse response and DTFT magnitude of the designed filter.

The figure depicts the impulse response resembling a sinc function and the DTFT exhibiting a window-like shape. Notably, the cutoff frequency is approximately 2, contributing to the filter's design characteristics.

VI. Filter Design Using the Kaiser Window

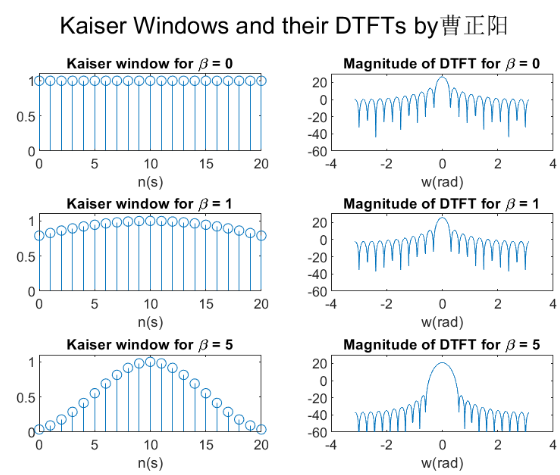

Fig. 13. Time domain plots and DTFTs of Kaiser windows for different β values.

- Effect of β on Window Shape:
  - Higher β values lead to narrower mainlobes, concentrating energy and improving potential frequency resolution.
  - However, higher β values also broaden the transition region, increasing spectral leakage.

- Effect of β on DTFT Sidelobes:
  - Increasing β typically results in lower sidelobe amplitudes, beneficial for reducing unwanted spectral components and improving stopband attenuation.
  - However, wider sidelobes with increasing β can lead to undesirable interactions with adjacent frequencies.

VII. Kaiser Window Design for Lowpass Filter

A. Computation of β and N

To design the Kaiser window for the lowpass filter, we need to calculate the values of β and N. The window function is defined as follows:

1) **Calculate A:**

$$A = -20 * log(\delta_s) = -20 * log(0.005) \approx 46.0205$$

2) **Calculate β:**

$$\beta = \begin{cases} 0.5842(A-21)^{0.4} + 0.07886(A-21) \\ \text{if } 21 \leq A \leq 50 \end{cases}$$

3) **Calculate N:**

$$N = \lceil 1 + \frac{A-8}{2.285(\omega_s - \omega_p)} \rceil$$

Find that $\beta = 4.0909$ and $N = 51$.

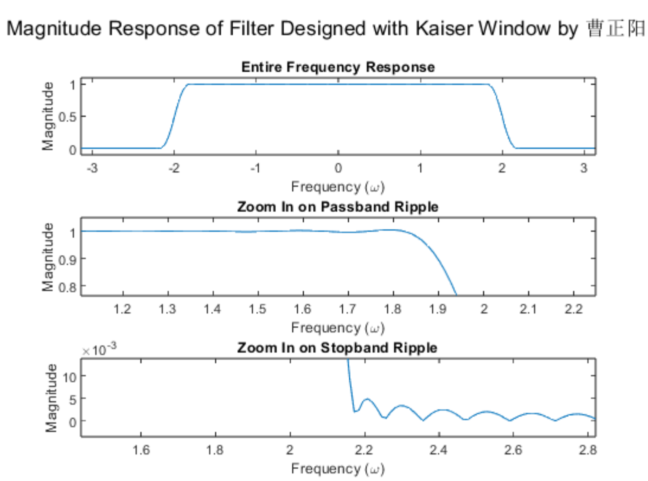

Fig. 14. Magnitude Response of the Kaiser Window Designed Filter.

Figure 14 shows the magnitude response of the Kaiser window designed filter. Three subplots provide an overview and zoomed-in views of passband and stopband ripples, considering design specifications.

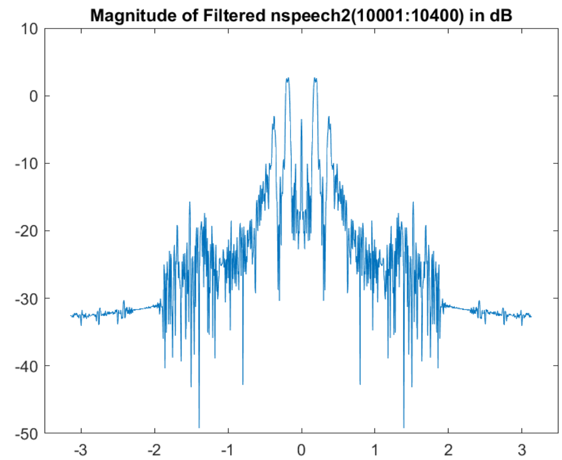

Fig. 15. Magnitude Response of the Filtered Signal.

Figure 15 illustrates the magnitude response of the filtered signal achieved through the application of the Kaiser window filter to the noisy speech signal. A comparative analysis with Figure 7.7 in the lab manual highlights the effective removal of noise from the signal.

Commenting on the impact of filtering on the frequency content and audio quality, it is observed that using a standard low-pass filter resulted in a reduction of noise but traces of background noise persisted. However, the application of the Kaiser window-designed filter proved highly effective. The filtered signal exhibited minimal residual noise, and the audio quality noticeably improved. A qualitative assessment suggests that if the listener can perceive the signal clearly, the filtering process with the Kaiser window design has been successful.

VIII. Conclusion

The lab experiments provided valuable insights into digital filter design. We explored the characteristics of FIR and IIR filters, observing the impact of parameters like θ and r on filter behavior. FIR filtering effectively removed specific frequency components from speech signals, showcasing improved audio quality.

The study of filter design using truncation highlighted trade-offs between filter size, stopband ripple, and computational complexity. Analysis of standard windows revealed key frequency domain characteristics, aiding in informed window selection.

The Kaiser window design for a lowpass filter demonstrated versatility in meeting design specifications, effectively removing noise from speech signals. These experiments contribute to our understanding of digital signal processing, emphasizing the importance of parameter selection and design methodologies for real-world applications.

IX. Code Listings

Here are listings of the source code provided.

A. lab7.m

```
%% 7.3 Design of a Simple FIR Filter
%%7.3.1
clear;
close all;

w = -pi:0.01:pi;
z = exp(1j * w);

H1 = 1 - 2 * cos(pi/6) * z.^(-1) + z
    .^(-2);
H2 = 1 - 2 * cos(pi/3) * z.^(-1) + z
    .^(-2);
H3 = 1 - 2 * cos(pi/2) * z.^(-1) + z
    .^(-2);
H4 = 1 - 2 * cos(2*pi/3) * z.^(-1) + z
    .^(-2);
H5 = 1 - 2 * cos(5*pi/6) * z.^(-1) + z
    .^(-2);
H6 = 1 - 2 * cos(pi) * z.^(-1) + z.^(-2);

figure;

subplot(2,3,1);
plot(w, abs(H1), 'LineWidth', 2);
title('\theta = \pi/6');
xlabel('\omega');
ylabel('|H(\omega)|');
grid on;
ylim([0 4.5]); % Set y-axis limits
xlim([-pi pi]); % Set x-axis limits

subplot(2,3,2);
plot(w, abs(H2), 'LineWidth', 2);
title('\theta = \pi/3');
xlabel('\omega');
ylabel('|H(\omega)|');
grid on;
ylim([0 4.5]); % Set y-axis limits
xlim([-pi pi]); % Set x-axis limits

subplot(2,3,3);
plot(w, abs(H3), 'LineWidth', 2);
title('\theta = \pi/2');
xlabel('\omega');
ylabel('|H(\omega)|');
grid on;
ylim([0 4.5]); % Set y-axis limits
xlim([-pi pi]); % Set x-axis limits

subplot(2,3,4);
plot(w, abs(H4), 'LineWidth', 2);
title('\theta = 2\pi/3');
xlabel('\omega');
ylabel('|H(\omega)|');
grid on;
ylim([0 4.5]); % Set y-axis limits
xlim([-pi pi]); % Set x-axis limits

subplot(2,3,5);
plot(w, abs(H5), 'LineWidth', 2);
title('\theta = 5\pi/6');
xlabel('\omega');
ylabel('|H(\omega)|');
grid on;
ylim([0 4.5]); % Set y-axis limits
xlim([-pi pi]); % Set x-axis limits

subplot(2,3,6);
plot(w, abs(H6), 'LineWidth', 2);
title('\theta = \pi');
xlabel('\omega');
ylabel('|H(\omega)|');
grid on;
ylim([0 4.5]); % Set y-axis limits
xlim([-pi pi]); % Set x-axis limits

sgtitle('Magnitude Response of the Filter
     for Different \theta Values by
             ');

% Save the figure as a PNG file
saveas(gcf, '7.3
   _filter_magnitude_response.png');


%% 7.3.2
clear;
close all;

% Load the audio signal and apply the FIR
    filter
load nspeech1;
filtered_signal = FIRfilter(nspeech1);

if 0 == 1
    sound(nspeech1)
else
    sound(filtered_signal)
end

% Extract segments for visualization
original_segment = nspeech1(100:200);
original_full = nspeech1(100:1100);
filtered_segment = filtered_signal
   (100:200);
filtered_full = filtered_signal(100:1100)
   ;

% Compute DTFT for visualization
```

```
[X_original, w_original] = DTFT(original_full, 0);
[X_filtered, w_filtered] = DTFT(filtered_full, 0);

% Set a common color
line_color = 'b-';
line_width = 0.8;  % Adjust the line width

% Plotting with a larger figure window size
figure('Position', [100, 100, 1000, 800]);

subplot(2, 2, 1);
plot(0:length(original_segment) - 1, original_segment, line_color, 'LineWidth', line_width);
title('Original Signal (Time Domain)');
xlabel('Sample Index');
ylabel('Amplitude');

subplot(2, 2, 2);
plot(w_original, 20 * log10(abs(X_original)), line_color, 'LineWidth', line_width);
title('Original Signal (Magnitude of DTFT in dB)');
xlabel('Frequency (rad/sample)');
ylabel('|X(w)| (dB)');

subplot(2, 2, 3);
plot(0:length(filtered_segment) - 1, filtered_segment, line_color, 'LineWidth', line_width);
title('Filtered Signal (Time Domain)');
xlabel('Sample Index');
ylabel('Amplitude');

subplot(2, 2, 4);
plot(w_filtered, 20 * log10(abs(X_filtered)), line_color, 'LineWidth', line_width);
title('Filtered Signal (Magnitude of DTFT in dB)');
xlabel('Frequency (rad/sample)');
ylabel('|X(w)| (dB)');

% Custom sgtitle
sgtitle('Audio Signal and FIR Filtering Analysis by           ');

% Save the figure with a different name
saveas(gcf, 'audio_signal_fir_filtering_analysis.png');
```

```
%% 7.4 Design of a Simple IIR Filter
%%7.4.1
clear;
close all;
% Frequency range
w = -pi:0.01:pi;
z = exp(1j * w);

% Define different values for r
r_values = [0.99, 0.9, 0.7, 0.5, 0.3];

% Initialize the subplot layout
figure('Position', [100, 100, 1200, 800]);

% Loop through each value of r
for i = 1:length(r_values)
    % Calculate Hi for the current r
    Hi = (1 - r_values(i)) ./ (1 - 2 * r_values(i) * cos(pi/3) * z.^(-1) + (r_values(i)^2) * (z.^(-2)));

    Hi_dB = 20 * log10(abs(Hi));
    % Create a subplot for each r value
    subplot(1, 5, i);
    plot(w, abs(Hi), 'LineWidth', 1.5);
    title(['r = ' num2str(r_values(i))]);
    xlabel('\omega');
    ylabel('|H_i(\omega)|');
    grid on;
    xlim([-pi pi]); % Set x-axis limits
    ylim([0 0.9])
end

sgtitle('Magnitude Response of H_i(z) for Different r Values by          ');
saveas(gcf, '7.4magnitude_response_plot.png');

%% 7.4.2
clear;
close all;
load pcm;

% Extracting segments for visualization
original_signal = pcm(100:200);
full_original_signal = pcm(100:1100);

% Time indices for plots
time_indices = 0:length(original_signal) - 1;

% Compute DTFT for the original signal
[X_original, freq_original] = DTFT(full_original_signal, 0);
```

```matlab
% Apply IIR filter to the signal
filtered_signal = IIRfilter(pcm);

if 0 == 1
    sound(pcm)
else
    sound(filtered_signal)
end

% Extract segments for visualization
filtered_segment = filtered_signal(100:200);
full_filtered_signal = filtered_signal(100:1100);

% Compute DTFT for the filtered signal
[X_filtered, freq_filtered] = DTFT(full_filtered_signal, 0);

% Angular frequency range around theta
theta = (3146 / 8000) * 2 * pi;
zoomed_freq_range = freq_original(freq_original > (theta - 0.02) & freq_original < (theta + 0.02));

% Extract corresponding values for zoomed DTFT
zoomed_original_DTFT = X_original(freq_original > (theta - 0.02) & freq_original < (theta + 0.02));
zoomed_filtered_DTFT = X_filtered(freq_original > (theta - 0.02) & freq_original < (theta + 0.02));

% Plotting
figure('Position', [100, 100, 1000, 800]);

subplot(3, 2, 1);
plot(time_indices, original_signal);
title('Original Signal (Time Domain)');
xlabel('Sample Index');
ylabel('Amplitude');
ylim([-0.1 0.1]);

subplot(3, 2, 2);
plot(freq_original, 20 * log10(abs(X_original)));
title('Original Signal (Magnitude of DTFT in dB)');
xlabel('Frequency (rad/sample)');
ylabel('|X(w)| (dB)');
xlim([-pi pi]);
ylim([-40 40]);
```

```matlab
subplot(3, 2, 3);
plot(time_indices, filtered_segment);
title('Filtered Signal (Time Domain)');
xlabel('Sample Index');
ylabel('Amplitude');
ylim([-0.1 0.1]);

subplot(3, 2, 4);
plot(freq_filtered, 20 * log10(abs(X_filtered)));
title('Filtered Signal (Magnitude of DTFT in dB)');
xlabel('Frequency (rad/sample)');
ylabel('|X(w)| (dB)');
xlim([-pi pi]);
ylim([-40 40]);

subplot(3, 2, 5);
stem(zoomed_freq_range, 20 * log10(abs(zoomed_original_DTFT)));
title('Zoomed DTFT of Original Signal');
xlabel('Frequency (rad/sample)');
ylabel('|X(w)| (dB)');
ylim([0 30]);

subplot(3, 2, 6);
stem(zoomed_freq_range, 20 * log10(abs(zoomed_filtered_DTFT)));
title('Zoomed DTFT of Filtered Signal');
xlabel('Frequency (rad/sample)');
ylabel('|X(w)| (dB)');
ylim([0 30]);

% Custom sgtitle
sgtitle('Audio Signal and IIR Filtering Analysis by            ');

% Save the figure with a different name
saveas(gcf, 'audio_signal_iir_filtering_analysis_with_r=0.995.png');

%% 7.6 Filter Design Using Truncation
clear;
close all;

% Load the noisy speech signal
load nspeech2;

% Cutoff frequency for the lowpass filter
cutoff_freq = 2.0;
% Define passband, transition band, and stopband limits
passband_limit = 1.8;
stopband_limit = 2.2;
```

```matlab
% Filter sizes
filter_size1 = 21;
filter_size2 = 101;

% Compute the filters
filter1 = LPFtrunc(filter_size1, cutoff_freq);
filter2 = LPFtrunc(filter_size2, cutoff_freq);

% Compute the magnitude responses
[magnitude_response1, frequency1] = DTFT(filter1, 512);
[magnitude_response2, frequency2] = DTFT(filter2, 512);

% Plot the magnitude responses (not in decibels)
figure;
subplot(2, 1, 1);
plot(frequency1, abs(magnitude_response1), 'LineWidth', 1.5);
title(['Magnitude Response - Filter Size: ' num2str(filter_size1)]);
xlabel('\omega');
ylabel('|H(e^{j\omega})|');
% Add annotations for passband, transition band, and stopband
hold on;
% Use drawrectangle to mark the passband, transition band, and stopband
rectangle('Position', [-passband_limit, 0.95, 2*passband_limit, 0.14], 'EdgeColor', 'r', 'LineStyle', '--'); % Passband
rectangle('Position', [passband_limit, 0, 2*(cutoff_freq-passband_limit)-0.02, 1.09], 'EdgeColor', 'b', 'LineStyle', '--');  % Transition band
rectangle('Position', [-passband_limit-2*(cutoff_freq-passband_limit)+0.02, 0, 2*(cutoff_freq-passband_limit)-0.02, 1.09], 'EdgeColor', 'b', 'LineStyle', '--');  % Transition band
rectangle('Position', [stopband_limit-0.02, 0, pi-stopband_limit+0.02, 0.1], 'EdgeColor', 'g', 'LineStyle', '--');  % Stopband
rectangle('Position', [-pi, 0, pi-stopband_limit+0.02, 0.1], 'EdgeColor', 'g', 'LineStyle', '--');  % Stopband
grid on;  % Add grid lines for better visualization

subplot(2, 1, 2);
plot(frequency2, abs(magnitude_response2), 'LineWidth', 1.5);
title(['Magnitude Response - Filter Size: ' num2str(filter_size2)]);
xlabel('\omega');
ylabel('|H(e^{j\omega})|');
% Add annotations for passband, transition band, and stopband
hold on;
rectangle('Position', [-passband_limit-0.14, 0.93, 2*passband_limit+0.28, 0.18], 'EdgeColor', 'r', 'LineStyle', '--');  % Passband
rectangle('Position', [1.93895, 0, 0.1, 1.11], 'EdgeColor', 'b', 'LineStyle', '--');  % Transition band
rectangle('Position', [-2.037, 0, 0.1, 1.11], 'EdgeColor', 'b', 'LineStyle', '--');  % Transition band
rectangle('Position', [2.04, 0, 1.1, 0.1], 'EdgeColor', 'g', 'LineStyle', '--');  % Stopband
rectangle('Position', [-pi, 0, pi-stopband_limit+0.166, 0.1], 'EdgeColor', 'g', 'LineStyle', '--');  % Stopband
grid on;  % Add grid lines for better visualization

% Save the figure as PNG
saveas(gcf, 'magnitude_response_plot_for_truncation_filter.png');

% Plot the magnitude responses (in decibels)
figure;
subplot(2, 1, 1);
plot(frequency1, 20*log(abs(magnitude_response1)), 'LineWidth', 1.5);
title(['Magnitude Response - Filter Size: ' num2str(filter_size1)]);
xlabel('\omega');
ylabel('|H(e^{j\omega})| (dB)');
grid on;  % Add grid lines for better visualization

subplot(2, 1, 2);
plot(frequency2, 20*log(abs(magnitude_response2)), 'LineWidth', 1.5);
title(['Magnitude Response - Filter Size: ' num2str(filter_size2)]);
xlabel('\omega');
ylabel('|H(e^{j\omega})| (dB)');
```

```matlab
grid on;  % Add grid lines for better visualization

% Save the figure as PNG
saveas(gcf, 'magnitude_response_plot_for_truncation_filter_in_dB.png');

% Convolve the filters with the noisy speech signal
filtered_signal1 = conv(nspeech2, filter1, 'same');
filtered_signal2 = conv(nspeech2, filter2, 'same');

% Adjust volume for better listening experience
filtered_signal1 = filtered_signal1 * 2;
filtered_signal2 = filtered_signal2 * 2;

% Listen to the unfiltered and filtered signals
sound(nspeech2);
pause(length(nspeech2)/8000);  % Pause to allow completion of the first sound
sound(filtered_signal1, 8000);
pause(length(filtered_signal1)/8000);  % Pause to allow completion of the second sound
sound(filtered_signal2, 8000);

%% 7.7 Filter Design Using Standard Windows
clear;
close all;

N = 0:20; % Extend the range to add more blank points
rectangular = ones(21, 1);
hanning = hann(21);
ham = hamming(21);
blackm = blackman(21);

[X1, w1] = DTFT(rectangular, 512);
[X2, w2] = DTFT(hanning, 512);
[X3, w3] = DTFT(ham, 512);
[X4, w4] = DTFT(blackm, 512);

H = ((2/pi) .* sinc((2/pi) .* (N - (0.5 * (length(N) - 1)))))';
w = H .* ham;
[X, W] = DTFT(w, 512);

% Figure 1: Time-domain representation of windows
figure;
subplot(221);
stem(N, rectangular, 'o-');
title('Rectangular window');
xlabel('Sample Index');
ylabel('Amplitude');
ylim([0 1.1]);

subplot(222);
plot(N, hanning, 'o-');
title('Hanning window');
xlabel('Sample Index');
ylabel('Amplitude');
ylim([0 1.1]);

subplot(223);
plot(N, ham, 'o-');
title('Hamming window');
xlabel('Sample Index');
ylabel('Amplitude');
ylim([0 1.1]);

subplot(224);
plot(N, blackm, 'o-');
title('Blackman window');
xlabel('Sample Index');
ylabel('Amplitude');
ylim([0 1.1]);

saveas(gcf, 'time_domain_windows.png');

% Figure 2: Frequency-domain representation of windows
figure;
subplot(221);
plot(w1, 20*log10(abs(X1)));
title('DTFT of Rectangular window');
xlabel('\omega');
ylabel('Magnitude (dB)');

subplot(222);
plot(w2, 20*log10(abs(X2)));
title('DTFT of Hanning window');
xlabel('\omega');
ylabel('Magnitude (dB)');

subplot(223);
plot(w3, 20*log10(abs(X3)));
title('DTFT of Hamming window');
xlabel('\omega');
ylabel('Magnitude (dB)');

subplot(224);
plot(w4, 20*log10(abs(X4)));
title('DTFT of Blackman window');
xlabel('\omega');
ylabel('Magnitude (dB)');
```

```
saveas(gcf, 'frequency_domain_windows.png');

% Figure 3: Filter Impulse response and DTFT
figure;
subplot(121);
plot(N, w, 'o-');
title('Filter Impulse response');
xlabel('Sample Index');
ylabel('Amplitude');

subplot(122);
plot(W, 20*log10(abs(X)));
title('Magnitude of Filter DTFT');
xlabel('\omega');
ylabel('Magnitude (dB)');

saveas(gcf, 'filter_impulse_response_and_dtft.png');



%% 7.8 Filter Design Using the Kaiser Window
clear;
close all;

% Time indices
n = 0:20;

% Generate Kaiser windows for different beta values
kaiserWindow1 = kaiser(21, 0);
kaiserWindow2 = kaiser(21, 1);
kaiserWindow3 = kaiser(21, 5);

% Compute the DTFTs for each Kaiser window
[X1, w1] = DTFT(kaiserWindow1, 512);
[X2, w2] = DTFT(kaiserWindow2, 512);
[X3, w3] = DTFT(kaiserWindow3, 512);

% Plot the Kaiser windows and their DTFTs
figure(1);
sgtitle('Kaiser Windows and their DTFTs b y ');
subplot(3, 2, 1);
stem(n, kaiserWindow1, 'o-');
xlabel('n(s)');
title('Kaiser window for \beta = 0');
ylim([0 1.1]);

subplot(3, 2, 2);
```

```
plot(w1, 20*log10(abs(X1)));
xlabel('w(rad)');
title('Magnitude of DTFT for \beta = 0');
ylim([-60 30]);

subplot(3, 2, 3);
stem(n, kaiserWindow2, 'o-');
xlabel('n(s)');
title('Kaiser window for \beta = 1');
ylim([0 1.1]);

subplot(3, 2, 4);
plot(w2, 20*log10(abs(X2)));
xlabel('w(rad)');
title('Magnitude of DTFT for \beta = 1');
ylim([-60 30]);

subplot(3, 2, 5);
stem(n, kaiserWindow3, 'o-');
xlabel('n(s)');
title('Kaiser window for \beta = 5');
ylim([0 1.1]);

subplot(3, 2, 6);
plot(w3, 20*log10(abs(X3)));
xlabel('w(rad)');
title('Magnitude of DTFT for \beta = 5');
ylim([-60 30]);


% Design a filter using a Kaiser window
figure(2);
load nspeech2;
omega_p = 1.8;
omega_c = 2.0;
omega_s = 2.2;
delta_p = 0.05;
delta_s = 0.005;
kaiserBeta = 4.0909;
filterOrder = 51;
kaiserWindow = kaiser(filterOrder, kaiserBeta);
filterImpulseResponse = LPFtrunc(filterOrder, 2);
filteredFilter = filterImpulseResponse .* kaiserWindow';
[H, omega] = DTFT(filteredFilter, 512);

% Plot the magnitude response of the filter
sgtitle('Magnitude Response of Filter Designed with Kaiser Window by ');

% Plot 1: Entire frequency response
subplot(3, 1, 1);
```

```
plot(omega, abs(H));
title('Entire Frequency Response');
xlabel('Frequency (\omega)');
ylabel('Magnitude');
xlim([-pi, pi]);
ylim([-0.1, 1.1]);

% Plot 2: Zoom in on passband ripple
subplot(3, 1, 2);
plot(omega, abs(H));
title('Zoom In on Passband Ripple');
xlabel('Frequency (\omega)');
ylabel('Magnitude');
xlim([-pi, pi]);
ylim([-0.1, 1.1]);
axis([omega_p-0.2, omega_p+0.2, 1-delta_p
    , 1+delta_p]);

% Plot 3: Zoom in on stopband ripple
subplot(3, 1, 3);
plot(omega, abs(H));
title('Zoom In on Stopband Ripple');
xlabel('Frequency (\omega)');
ylabel('Magnitude');
xlim([-pi, pi]);
ylim([-0.1, 1.1]);
axis([omega_s-0.2, omega_s+0.2, 0,
    delta_s]);


% Apply the filter to the noisy speech
    signal
filteredSpeech = conv(filteredFilter,
    nspeech2);
[filteredDTFT, filteredOmega] = DTFT(
    filteredSpeech(10001:10400), 1024);

% Plot the magnitude response of the
    filtered signal
figure(3);
plot(filteredOmega, 20*log10(abs(
    filteredDTFT)));
title('Magnitude of Filtered nspeech2
    (10001:10400) in dB');
xlim([-3.5 3.5]);

% Save all figures as PNG
saveas(figure(1), '
    kaiser_windows_and_dtfts.png');
saveas(figure(2), '
    magnitude_response_kaiser_filter.png')
    ;
saveas(figure(3), '
    magnitude_response_filtered_signal.png
    ');
```

```
% Play the sounds
sound(nspeech2, 8e3);
pause(length(nspeech2)/8e3);  % Pause to
    listen to the original sound
sound(filteredSpeech, 8e3);


%% FUNCTIONS
function y = FIRfilter(x)
    % Compute DTFT
    [X, w] = DTFT(x, 0);
    % Find the index of the maximum
       magnitude in the DTFT
    [~, Imax] = max(abs(X));
    % Design the FIR filter
    theta = w(Imax);
    h = [1 -2*cos(theta) 1];
    % Apply the FIR filter using
       convolution
    y = conv(x, h, 'same');
end

function y = IIRfilter(x)
    % Parameters
    theta = (3146 / 8000) * 2 * pi;
    r = 0.995;
    % Length of input signal
    N = length(x);
    % Initialize output signal
    y = zeros(1, N);
    % Apply recursive difference equation
    y(1) = (1-r)*x(1);
    y(2) = (1-r)*x(2) + 2 * r * cos(theta
       )*y(1);
    for i = 3:N
        y(i) = (1-r)*x(i) + 2 * r * cos(
            theta) * y(i - 1) - (r^2) * y(
            i - 2);
    end
end
function h = LPFtrunc(N, wc)
    % LPFtrunc: Compute the truncated and
        shifted impulse response of a
       lowpass filter
    % Check if N is even, if so, make it
       odd
    if mod(N, 2) == 0
        N = N + 1;
    end

    % Compute the truncated and shifted
       impulse response
    n = -(N-1)/2:(N-1)/2;
    h = wc/pi * sinc(wc * n / pi);
end
```

B. DTFT.m

```
function [X,w] = DTFT(x,M)
% This function computes samples of the
   DTFT of x.
% To compute the DTFT of x, use
%
%              [X,w] = DTFT(x,0)
%
% where X is the vector of DTFT samples
   and w is the
% vector of radial frequencies. To
   compute at least
% M samples of the DTFT, you may use the
   command
%
%              [X,w] = DTFT(x,M)
%
% This is useful when the plot of X
   versus w does
% not contain a sufficient number of
   points.

N = max(M,length(x));
N = 2^(ceil(log(N)/log(2)));

% Take the padded fft
X = fft(x,N);
w = 2*pi*( (0:(N-1))/N );
w = w - 2*pi*(w>=pi);

% Shift FFT to go from -pi to pi
X = fftshift(X);
w = fftshift(w);
```