Contents

- QUESTION 1 COMMENTING
- QUESTION 2: DTFT OF COMMON FUNCTIONS
- 2 (a) PLOT DTFT
- 2 (b) PLOT DTFT
- 2 (c) PLOT DTFT
- 2 (d) PLOT DTFT
- 2 (e) PLOT DTFT
- 2 (f) PLOT DTFT
- QUESTION 3: DTFT PROPERTIES
- 3(a) PLOT DTFT
- 3(b) PLOT DTFT
- 3(c) PLOT DTFT
- 3(d) PLOT DTFT
- 3(e) PLOT DTFT
- QUESTION 4: NULLING FILTER
- 4(a) EVALUATE DTFT OF INPUT SIGNAL
- 4(b) IDENTIFY FREQUENCY
- 4(c) DESIGN FILTER
- 4(d) APPLY FILTER
- 4(e) LISTEN TO AUDIO
- 4(f) EXTEND YOUR KNOWLEDGE TO NOISY2.WAV
- ALL FUNCTIONS SUPPORTING THIS CODE

QUESTION 1 COMMENTING

```
% DO NOT REMOVE THE LINE BELOW
% MAKE SURE 'eel3135_lab06_comment.m' IS IN THE SAME DIRECTORY AS THIS FILE
clear; close all; clc;
type('eel3135_lab06_comment.m')
```

```
%% USER-DEFINED VARIABLES
clear
close all
clc

%% DEFINE FILTER
N = 10;                                                                      % Defines the length of the filter
h = (1/N)*ones(N,1);                                                         % Create a moving average filter (box filter) of length N

% <-- Answer Question: What is the impulse response of this filter?
%     Use d in place of delta.
%
% The impulse response of this filter is h[n] = (1/N) * d[n] + (1/N) * d[n-1] + ... + (1/N) * d[n-(N-1)],
% where d[n] is the discrete-time impulse function. This means that the filter averages the last N samples.

% COMPUTE THE DTFT
n = 0:(N-1);                                                                 % Defines Time index of filter
w = -pi:pi/5000:pi;                                                          % Define the frequency range for DTFT computation
H = DTFT(h,w);                                                               % Compute the Discrete-Time Fourier Transform (DTFT) of the filter

% PLOT THE IMPULSE RESPONSE AND DTFT
figure
```

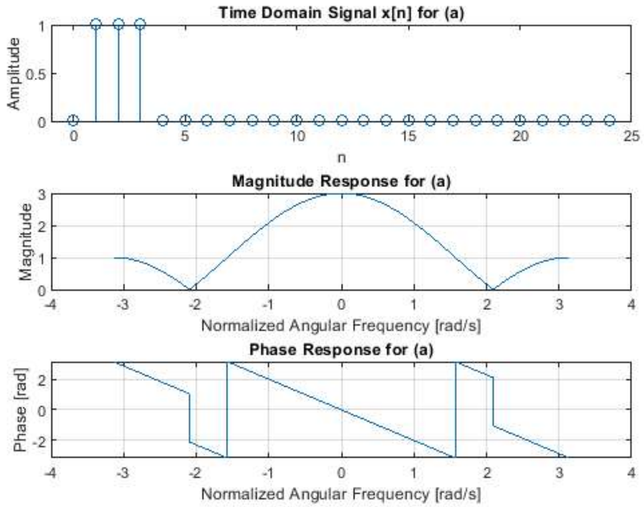
```
subplot(3,1,1)
stem(n,h)
xlim([-0.5 20.5])
title('Impulse Response of h')
xlabel('Time Index (n)')
ylabel('Amplitude')
subplot(3,1,2)
plot(w,abs(H))
grid on;
title('Magnitude Response of H')
ylabel('Magnitude [rad]')
xlabel('Normalized Angular Frequency [rad/s]')
subplot(3,1,3)
plot(w,angle(H))
grid on;
title('Phase Response of H')
ylabel('Phase [rad]')
xlabel('Normalized Angular Frequency [rad/s]')


function H = DTFT(x,w)
%  ===>
% This function computes the Discrete-Time Fourier Transform (DTFT) of a discrete-time signal x
% for a given set of angular frequencies w. The DTFT is calculated using the formula:
% H(e^jw) = sum(x[n] * e^(-jwn)), where n is the index of the signal x.
% <===

    H = zeros(length(w),1);                                         % Initialize the DTFT result vector
    for nn = 1:length(x)                                            % Loop over each sample in the input signal
        H = H + x(nn).*exp(-1j*w.'*(nn-1));                          % Accumulate the DTFT contributions
    end

end
```

QUESTION 2: DTFT OF COMMON FUNCTIONS

2 (a) PLOT DTFT

```
w = -pi:pi/5000:pi;                                                 % Define the frequency range for DTFT
n = 0:24;                                                           % Time index
x_a = [0, 1, 1, 1, zeros(1, 21)];                                   % x[n] = δ[n − 1] + δ[n − 2] + δ[n − 3]
H_a = DTFT(x_a, w);

% Plotting
figure;

subplot(3,1,1);
stem(n, x_a);
title('Time Domain Signal x[n] for (a)');
xlabel('n');
ylabel('Amplitude');
xlim([-1 25]);

subplot(3,1,2);
plot(w, abs(H_a));
title('Magnitude Response for (a)');
xlabel('Normalized Angular Frequency [rad/s]');
ylabel('Magnitude');
grid on;

subplot(3,1,3);
plot(w, angle(H_a));
title('Phase Response for (a)');
xlabel('Normalized Angular Frequency [rad/s]');
ylabel('Phase [rad]');
```

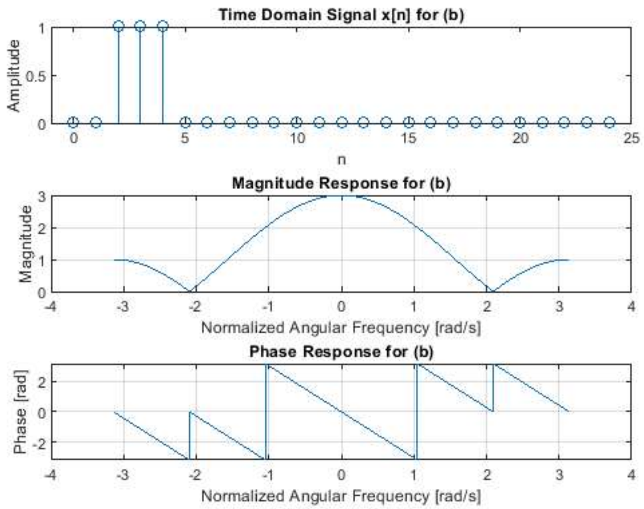
```
grid on;

% ALSO ANSWER: Is the data predominantly low frequency, high frequency,
%              or neither?
%
% The data is predominantly low frequency.
```



2 (b) PLOT DTFT

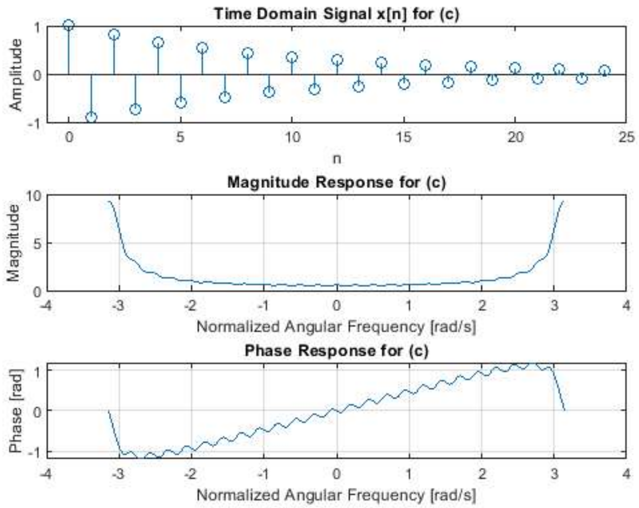
```
w = -pi:pi/5000:pi;                                      % Define the frequency range for DTFT
n = 0:24;                                                % Time index
x_b = [0, 0, 1, 1, 1, zeros(1, 20)];                     % x[n] = δ[n − 2] + δ[n − 3] + δ[n − 4]
H_b = DTFT(x_b, w);

% Plotting
subplot(3,1,1);
stem(n, x_b);
title('Time Domain Signal x[n] for (b)');
xlabel('n');
ylabel('Amplitude');
xlim([-1 25]);

subplot(3,1,2);
plot(w, abs(H_b));
title('Magnitude Response for (b)');
xlabel('Normalized Angular Frequency [rad/s]');
ylabel('Magnitude');
grid on;

subplot(3,1,3);
plot(w, angle(H_b));
title('Phase Response for (b)');
xlabel('Normalized Angular Frequency [rad/s]');
ylabel('Phase [rad]');
grid on;

% ALSO ANSWER: Is the data predominantly low frequency, high frequency,
%              or neither?
```

```
%
% The data is mainly low frequncy
```



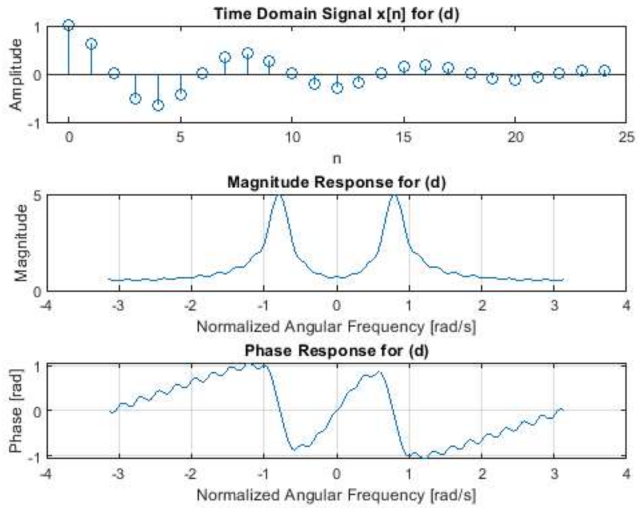
2 (c) PLOT DTFT

```
w = -pi:pi/5000:pi;                                                        % Define the frequency range for DTFT
n = 0:24;                                                                  % Time index
x_c = (-0.9).^n;                                                           % x[n] = (-0.9)^n * u[n]
x_c(n < 0) = 0;                                                            % Apply u[n]
H_c = DTFT(x_c, w);

% Plotting
subplot(3,1,1);
stem(n, x_c);
title('Time Domain Signal x[n] for (c)');
xlabel('n');
ylabel('Amplitude');
xlim([-1 25]);

subplot(3,1,2);
plot(w, abs(H_c));
title('Magnitude Response for (c)');
xlabel('Normalized Angular Frequency [rad/s]');
ylabel('Magnitude');
grid on;

subplot(3,1,3);
plot(w, angle(H_c));
title('Phase Response for (c)');
xlabel('Normalized Angular Frequency [rad/s]');
ylabel('Phase [rad]');
grid on;

% ALSO ANSWER: Is the data predominantly low frequency, high frequency,
%              or neither?
%
% The data is prdominantly low frequency
```



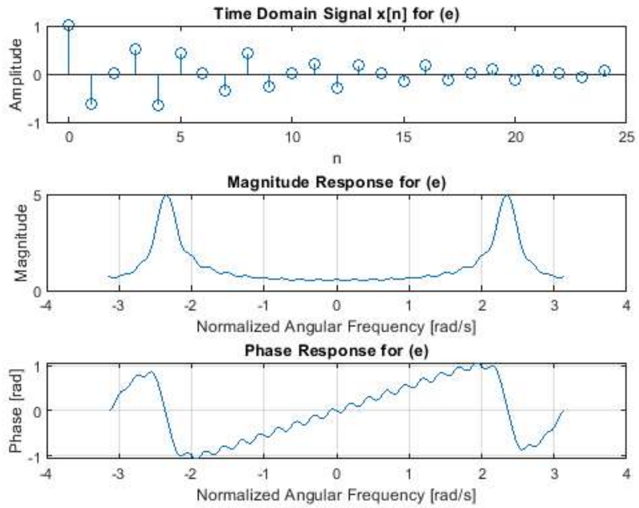
2 (d) PLOT DTFT

```
w = -pi:pi/5000:pi;                                         % Define the frequency range for DTFT
n = 0:24;                                                   % Time index
x_d = (-0.9).^n .* cos((3*pi/4)*n);                         % x[n] = (-0.9)^n * cos((3π/4)n) * u[n]
x_d(n < 0) = 0;                                             % Apply u[n]
H_d = DTFT(x_d, w);

% Plotting
subplot(3,1,1);
stem(n, x_d);
title('Time Domain Signal x[n] for (d)');
xlabel('n');
ylabel('Amplitude');
xlim([-1 25]);

subplot(3,1,2);
plot(w, abs(H_d));
title('Magnitude Response for (d)');
xlabel('Normalized Angular Frequency [rad/s]');
ylabel('Magnitude');
grid on;

subplot(3,1,3);
plot(w, angle(H_d));
title('Phase Response for (d)');
xlabel('Normalized Angular Frequency [rad/s]');
ylabel('Phase [rad]');
grid on;

% ALSO ANSWER: Is the data predominantly low frequency, high frequency,
%              or neither?
%
% The answer is predominantly neither
```



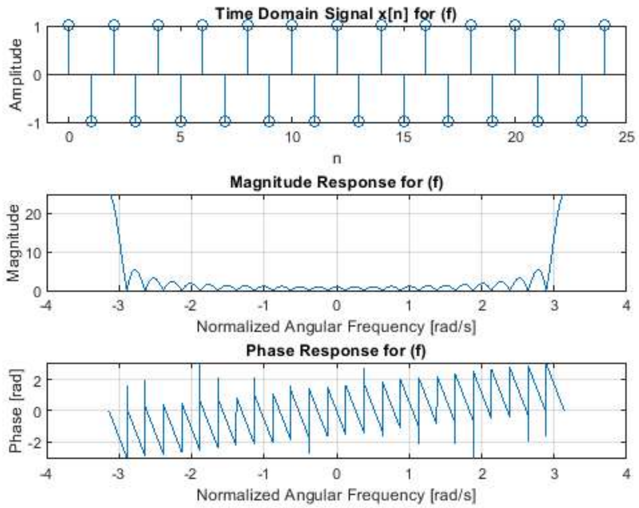
2 (e) PLOT DTFT

```
w = -pi:pi/5000:pi;                                % Define the frequency range for DTFT
n = 0:24;                                          % Time index
x_e = (-0.9).^n .* cos((pi/4)*n);                  % x[n] = (-0.9)^n * cos((π/4)n) * u[n]
x_e(n < 0) = 0;                                    % Apply u[n]
H_e = DTFT(x_e, w);

% Plotting
subplot(3,1,1);
stem(n, x_e);
title('Time Domain Signal x[n] for (e)');
xlabel('n');
ylabel('Amplitude');
xlim([-1 25]);

subplot(3,1,2);
plot(w, abs(H_e));
title('Magnitude Response for (e)');
xlabel('Normalized Angular Frequency [rad/s]');
ylabel('Magnitude');
grid on;

subplot(3,1,3);
plot(w, angle(H_e));
title('Phase Response for (e)');
xlabel('Normalized Angular Frequency [rad/s]');
ylabel('Phase [rad]');
grid on;

% ALSO ANSWER: Is the data predominantly low frequency, high frequency,
%              or neither?
%
% The answer is predominantly low frequency
```



2 (f) PLOT DTFT

```
w = -pi:pi/5000:pi;                                                         % Define the frequency range for DTFT
n = 0:24;                                                                   % Time index
x_f = (-1).^n;                                                              % x[n] = (-1)^n * u[n]
x_f(n < 0) = 0;                                                             % Apply u[n]
H_f = DTFT(x_f, w);

% Plotting
subplot(3,1,1);
stem(n, x_f);
title('Time Domain Signal x[n] for (f)');
xlabel('n');
ylabel('Amplitude');
xlim([-1 25]);

subplot(3,1,2);
plot(w, abs(H_f));
title('Magnitude Response for (f)');
xlabel('Normalized Angular Frequency [rad/s]');
ylabel('Magnitude');
grid on;

subplot(3,1,3);
plot(w, angle(H_f));
title('Phase Response for (f)');
xlabel('Normalized Angular Frequency [rad/s]');
ylabel('Phase [rad]');
grid on;

% ALSO ANSWER: Is the data predominantly low frequency, high frequency,
%               or neither?
%
% The frequency is predominantly high frequency
```

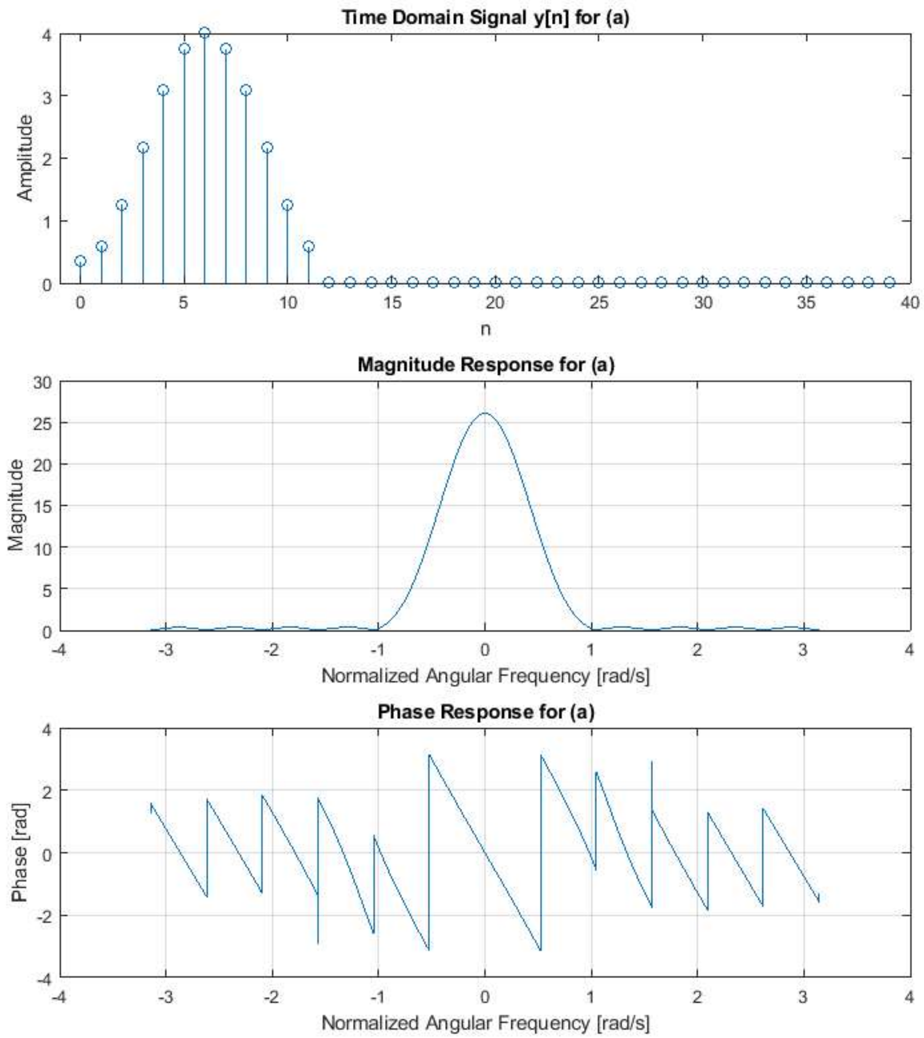


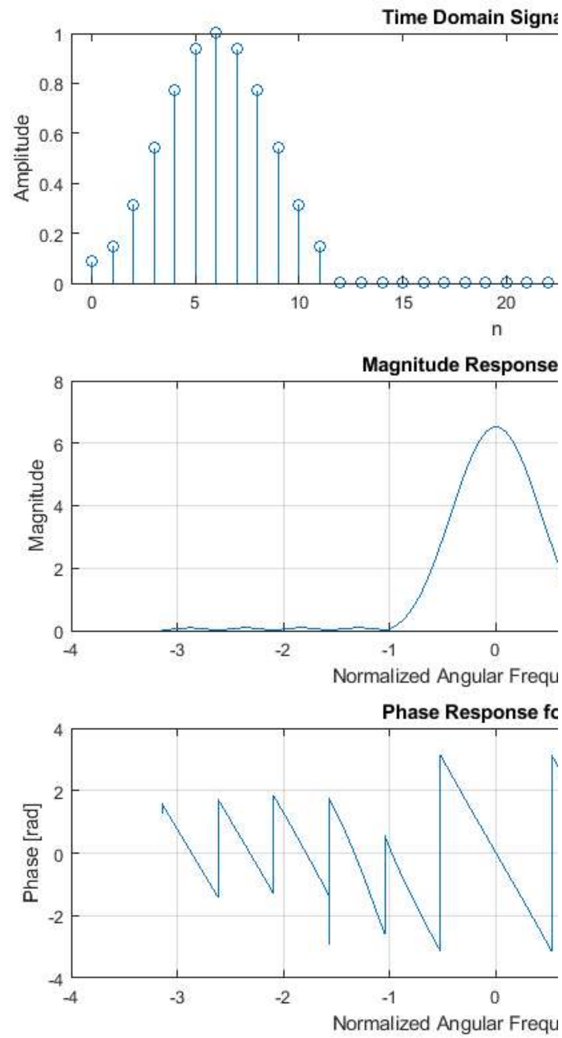
QUESTION 3: DTFT PROPERTIES

3(a) PLOT DTFT

```
n = 0:39;                                                               % Time index
x_q3 = (25/46) - (21/46) * cos((pi/6) * n);                            % x[n] = [(25/46) − (21/46) cos((π/6)n)] * (u[n] − u[n − 12])
x_q3(n >= 12) = 0;                                                      % Apply the window u[n] - u[n-12]
H_q3 = DTFT(x_q3, w);

y_a = 4 * x_q3;                                                         % y[n] = 4x[n]
H_y_a = DTFT(y_a, w);

% Plotting
figure('Units', 'normalized', 'OuterPosition', [0, 0, 1, 1]);          % Full screen

subplot(3,2,2);
stem(n, x_q3);
title('Time Domain Signal x[n]');
xlabel('n');
ylabel('Amplitude');
xlim([-1 40]);

subplot(3,2,4);
plot(w, abs(H_q3));
title('Magnitude Response for x[n]');
xlabel('Normalized Angular Frequency [rad/s]');
ylabel('Magnitude');
grid on;

subplot(3,2,6);
plot(w, angle(H_q3));
title('Phase Response for x[n]');
xlabel('Normalized Angular Frequency [rad/s]');
ylabel('Phase [rad]');
grid on;

subplot(3,2,1);
stem(n, y_a);
title('Time Domain Signal y[n] for (a)');
```

```
xlabel('n');
ylabel('Amplitude');
xlim([-1 40]);

subplot(3,2,3);
plot(w, abs(H_y_a));
title('Magnitude Response for (a)');
xlabel('Normalized Angular Frequency [rad/s]');
ylabel('Magnitude');
grid on;

subplot(3,2,5);
plot(w, angle(H_y_a));
title('Phase Response for (a)');
xlabel('Normalized Angular Frequency [rad/s]');
ylabel('Phase [rad]');
grid on;

% ALSO ANSWER: describe how each system changes the frequency domain
%
% The magnitude is scaled by a factor of 4, while the phase remains unchanged.
```

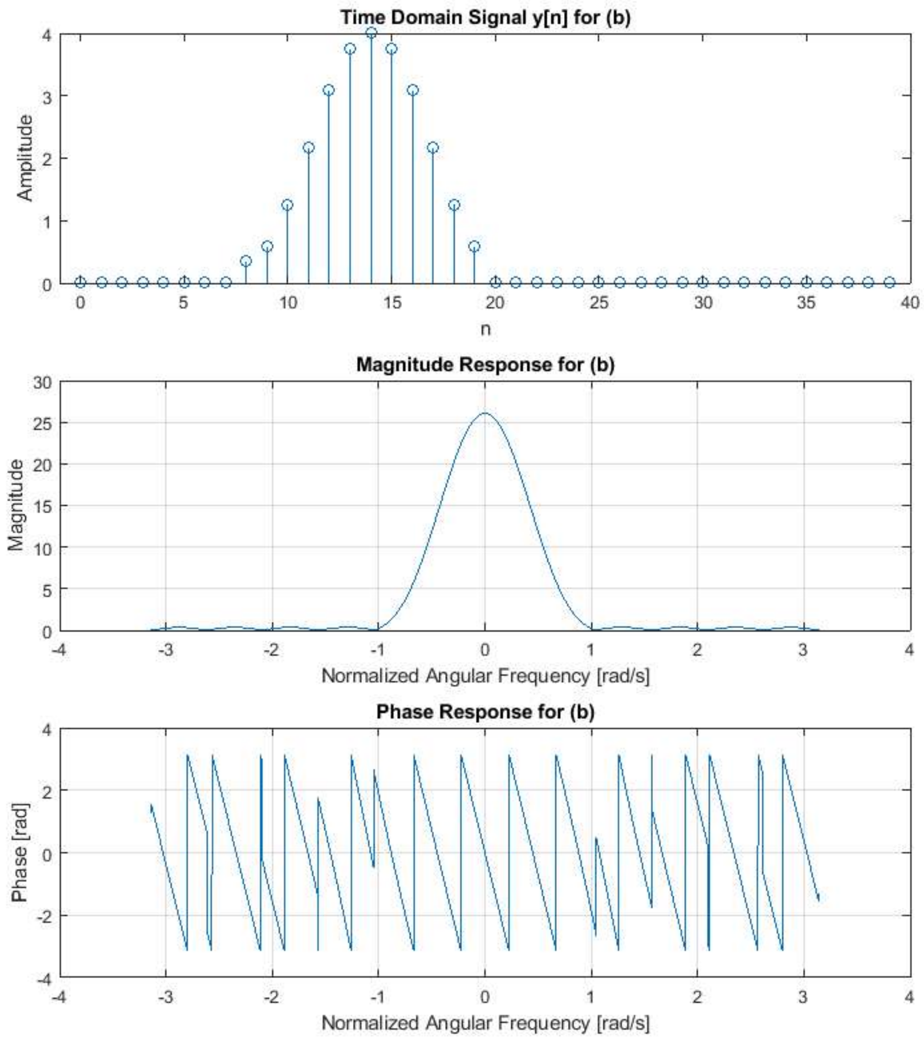


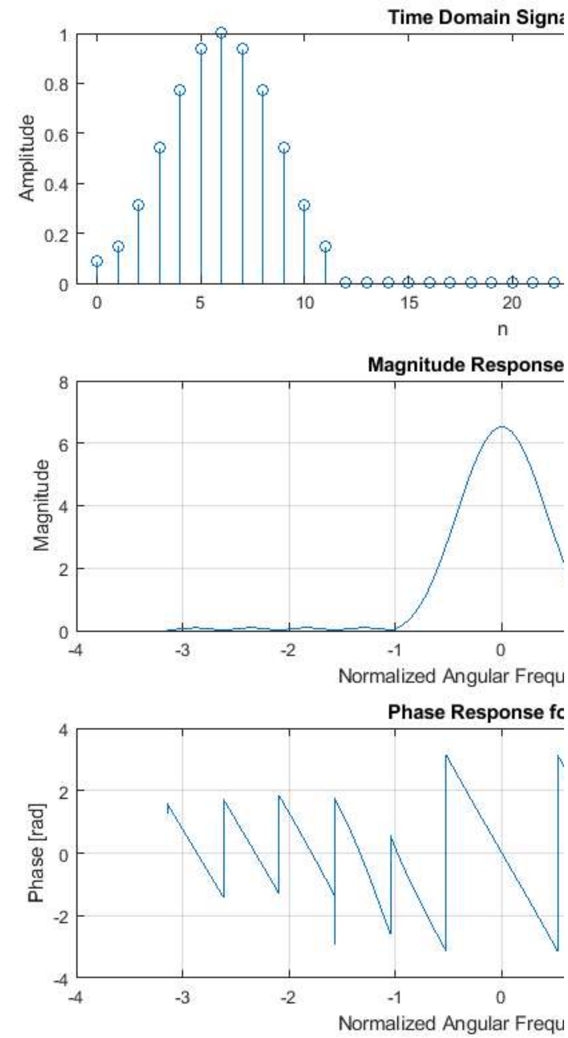


3(b) PLOT DTFT

```
n = 0:39;                                                  % Time index
x_q3 = (25/46) - (21/46) * cos((pi/6) * n);                % x[n] = [(25/46) − (21/46) cos((π/6)n)] * (u[n] − u[n − 12])
x_q3(n >= 12) = 0;                                         % Apply the window u[n] - u[n-12]
H_q3 = DTFT(x_q3, w);

y_b = 4 * x_q3;                                            % y[n] = 4x[n - 8]
y_b = [zeros(1, 8), y_b(1:end-8)];                         % Shift the signal
H_y_b = DTFT(y_b, w);

% Plotting
```

```
subplot(3,2,1);
stem(n, y_b);
title('Time Domain Signal y[n] for (b)');
xlabel('n');
ylabel('Amplitude');
xlim([-1 40]);

subplot(3,2,3);
plot(w, abs(H_y_b));
title('Magnitude Response for (b)');
xlabel('Normalized Angular Frequency [rad/s]');
ylabel('Magnitude');
grid on;

subplot(3,2,5);
plot(w, angle(H_y_b));
title('Phase Response for (b)');
xlabel('Normalized Angular Frequency [rad/s]');
ylabel('Phase [rad]');
grid on;

% ALSO ANSWER: describe how each system changes the frequency domain
%
% The magnitude is scaled by a factor of 4 and the phase is delayed by the shift.
```

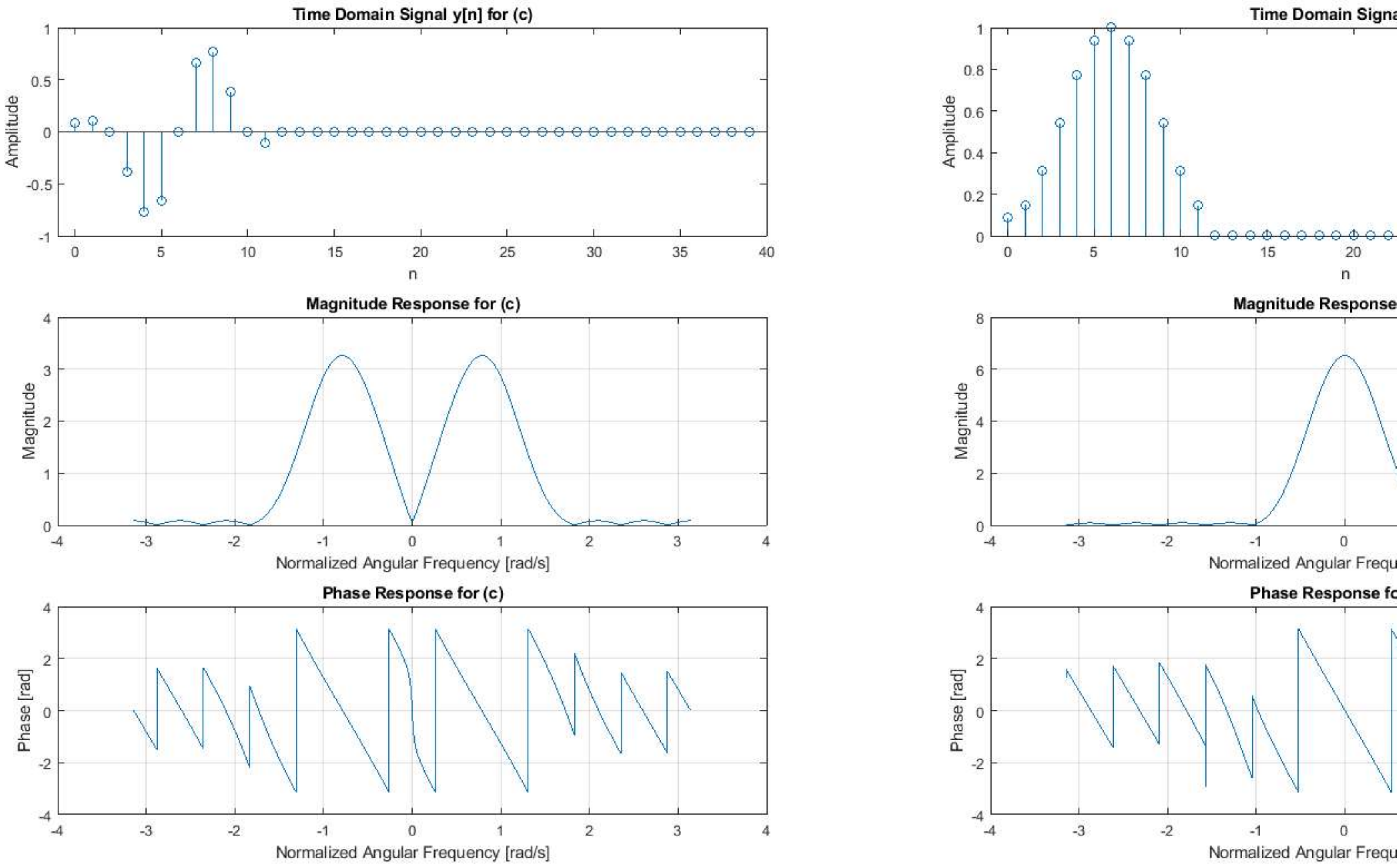




3(c) PLOT DTFT

```
n = 0:39;                                                   % Time index
x_q3 = (25/46) - (21/46) * cos((pi/6) * n);                 % x[n] = [(25/46) − (21/46) cos((π/6)n)] * (u[n] − u[n − 12])
x_q3(n >= 12) = 0;                                          % Apply the window u[n] - u[n-12]
H_q3 = DTFT(x_q3, w);

y_c = x_q3 .* cos((pi/4) * n);                              % y[n] = x[n] cos((π/4)n)
H_y_c = DTFT(y_c, w);

% Plotting
subplot(3,2,1);
```

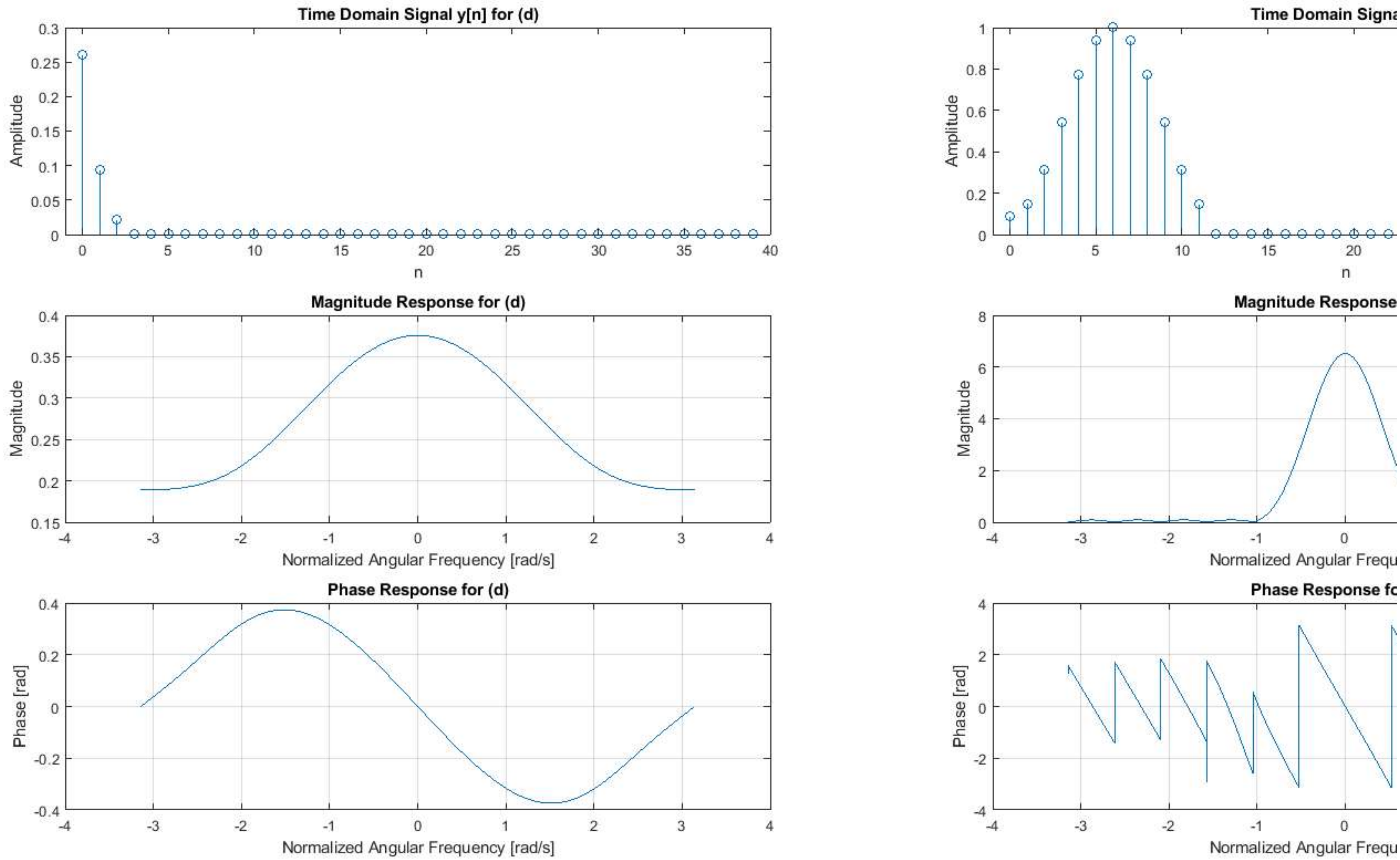
```
stem(n, y_c);
title('Time Domain Signal y[n] for (c)');
xlabel('n');
ylabel('Amplitude');
xlim([-1 40]);

subplot(3,2,3);
plot(w, abs(H_y_c));
title('Magnitude Response for (c)');
xlabel('Normalized Angular Frequency [rad/s]');
ylabel('Magnitude');
grid on;

subplot(3,2,5);
plot(w, angle(H_y_c));
title('Phase Response for (c)');
xlabel('Normalized Angular Frequency [rad/s]');
ylabel('Phase [rad]');
grid on;

% ALSO ANSWER: describe how each system changes the frequency domain
%
% The magnitude response is modulated by the cosine term, affecting the frequency content.
```



3(d) PLOT DTFT

```
n = 0:39;                                                   % Time index
x_q3 = (25/46) - (21/46) * cos((pi/6) * n);                 % x[n] = [(25/46) - (21/46) cos((π/6)n)] * (u[n] - u[n - 12])
x_q3(n >= 12) = 0;                                          % Apply the window u[n] - u[n-12]
H_q3 = DTFT(x_q3, w);

y_d = conv(x_q3, x_q3, 'same');                             % y[n] = x[n] * x[n] (convolution)
H_y_d = DTFT(y_d, w);

% Plotting
subplot(3,2,1);
```

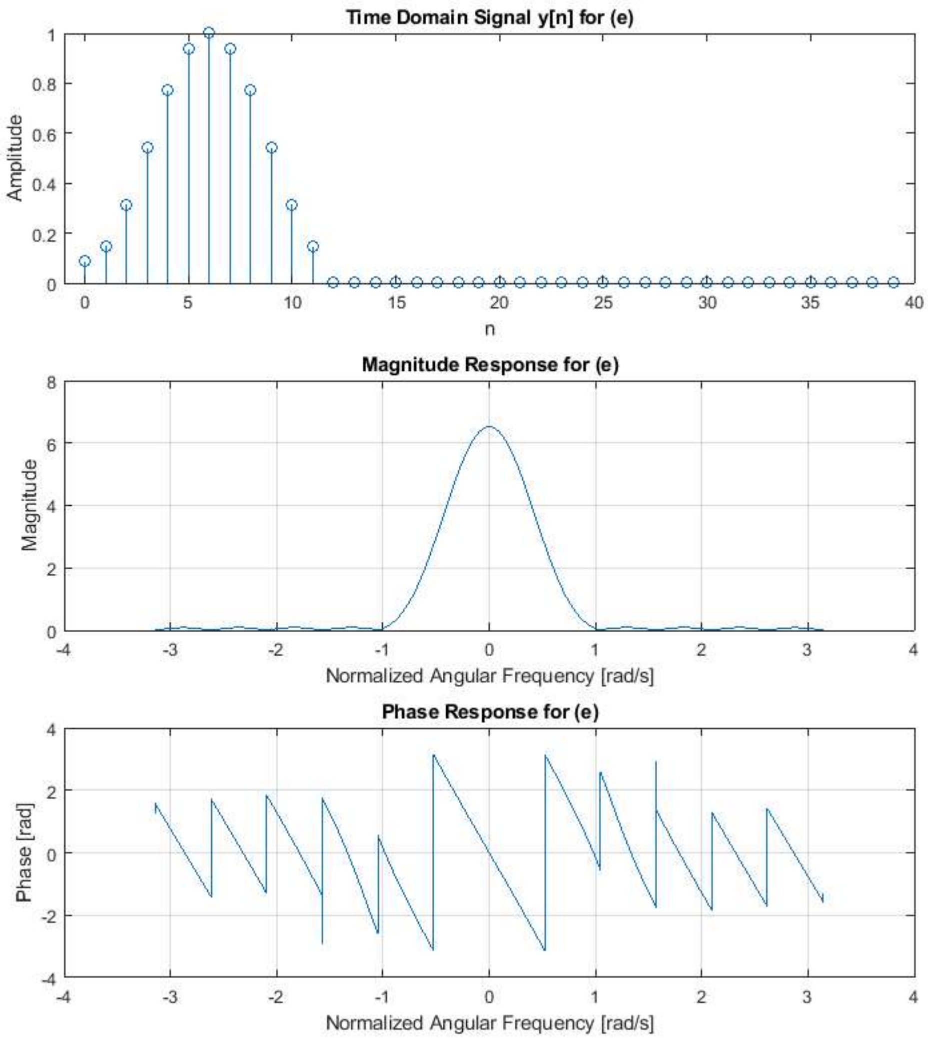
```
stem(n, y_d);
title('Time Domain Signal y[n] for (d)');
xlabel('n');
ylabel('Amplitude');
xlim([-1 40]);

subplot(3,2,3);
plot(w, abs(H_y_d));
title('Magnitude Response for (d)');
xlabel('Normalized Angular Frequency [rad/s]');
ylabel('Magnitude');
grid on;

subplot(3,2,5);
plot(w, angle(H_y_d));
title('Phase Response for (d)');
xlabel('Normalized Angular Frequency [rad/s]');
ylabel('Phase [rad]');
grid on;

% ALSO ANSWER: describe how each system changes the frequency domain
%
% The convolution increases the bandwidth of the signal, resulting in a wider magnitude response.
```

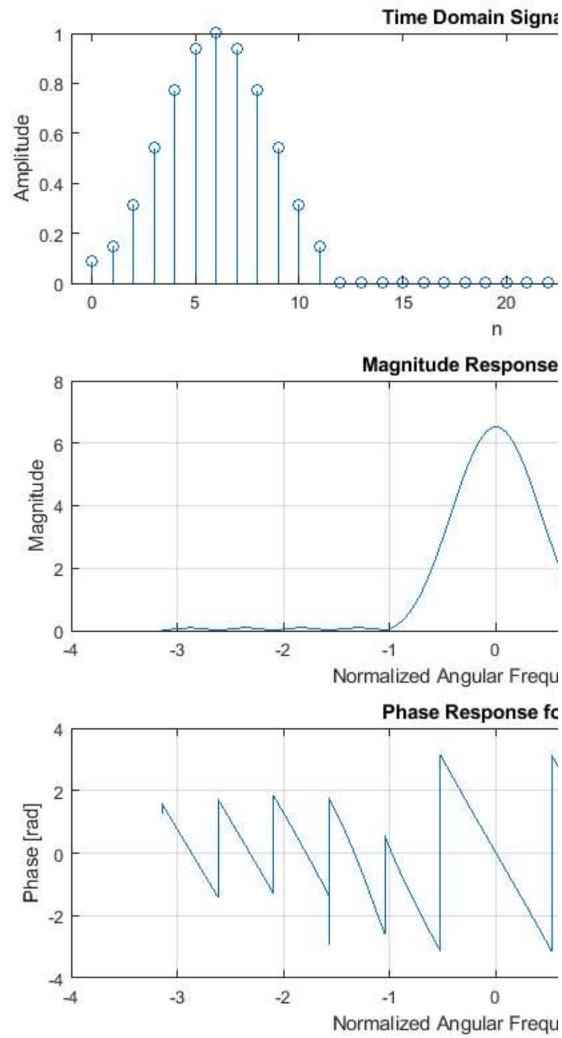


3(e) PLOT DTFT

```
n = 0:39;                                          % Time index
x_q3 = (25/46) - (21/46) * cos((pi/6) * n);        % x[n] = [(25/46) − (21/46) cos((π/6)n)] * (u[n] − u[n − 12])
x_q3(n >= 12) = 0;                                 % Apply the window u[n] - u[n-12]
H_q3 = DTFT(x_q3, w);

y_e = abs(x_q3); % y[n] = |x[n]|
H_y_e = DTFT(y_e, w);

% Plotting
subplot(3,2,1);
```

```
stem(n, y_e);
title('Time Domain Signal y[n] for (e)');
xlabel('n');
ylabel('Amplitude');
xlim([-1 40]);

subplot(3,2,3);
plot(w, abs(H_y_e));
title('Magnitude Response for (e)');
xlabel('Normalized Angular Frequency [rad/s]');
ylabel('Magnitude');
grid on;

subplot(3,2,5);
plot(w, angle(H_y_e));
title('Phase Response for (e)');
xlabel('Normalized Angular Frequency [rad/s]');
ylabel('Phase [rad]');
grid on;

% ALSO ANSWER: describe how each system changes the frequency domain
%
% The magnitude response is similar to the original signal, but the phase is not defined for non-negative values.
```





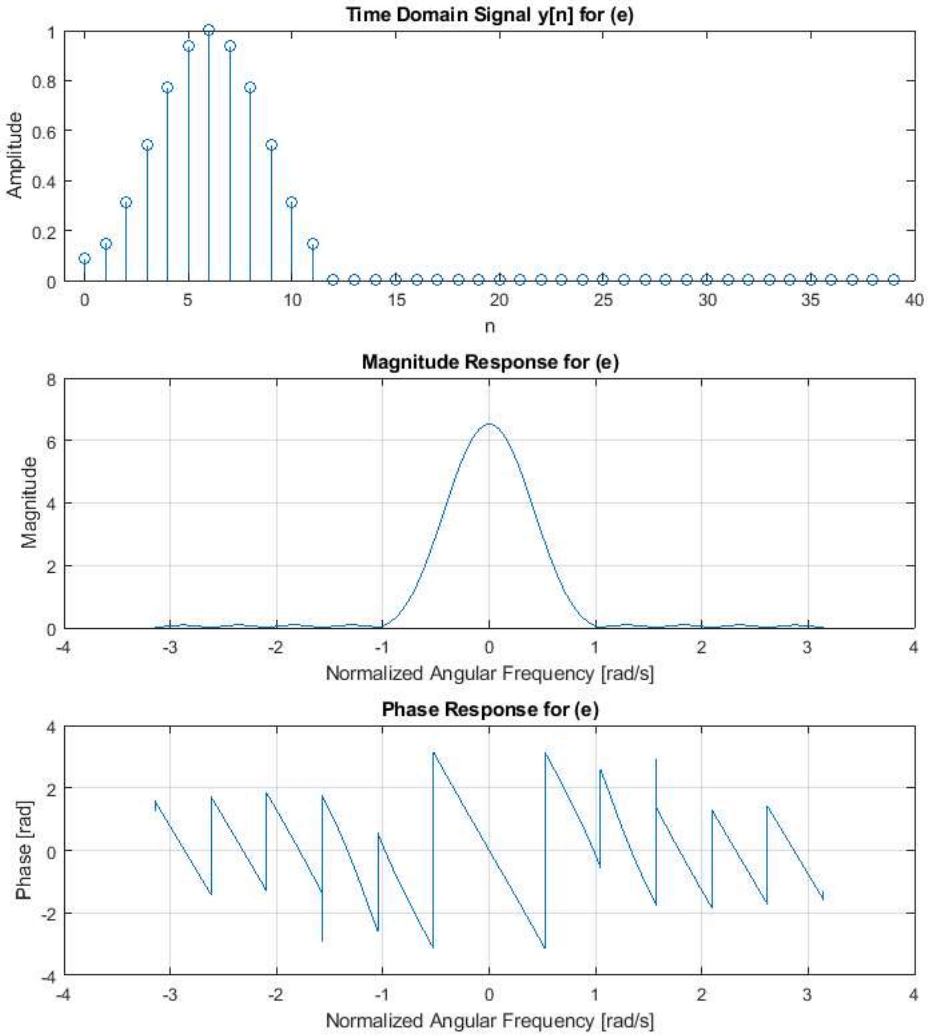
QUESTION 4: NULLING FILTER

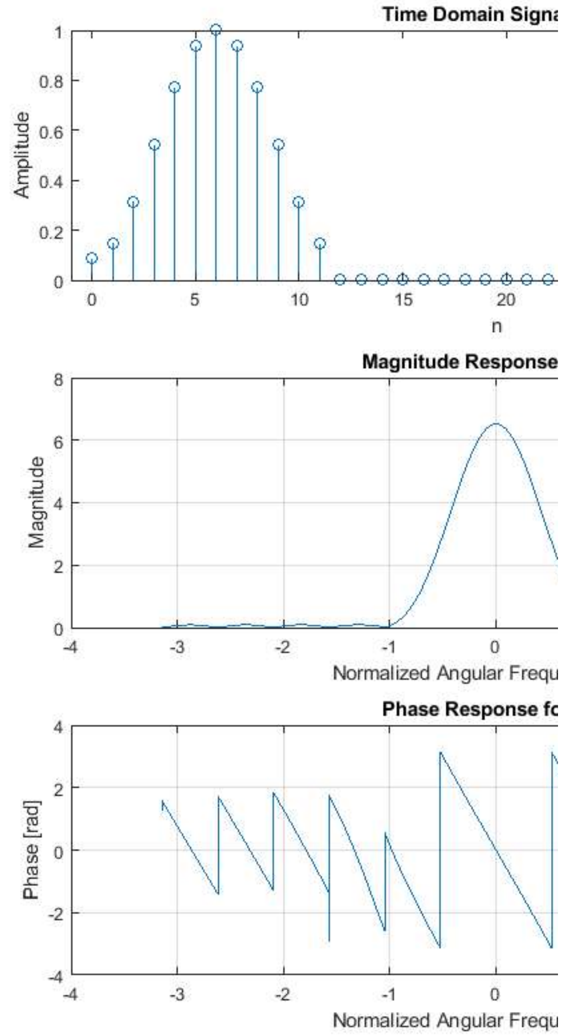
```
% DO NOT REMOVE THE LINE BELOW
% MAKE SURE 'noisy.wav' IS IN THE SAME DIRECTORY AS THIS FILE
[x, fs] = audioread('noisy.wav');
```

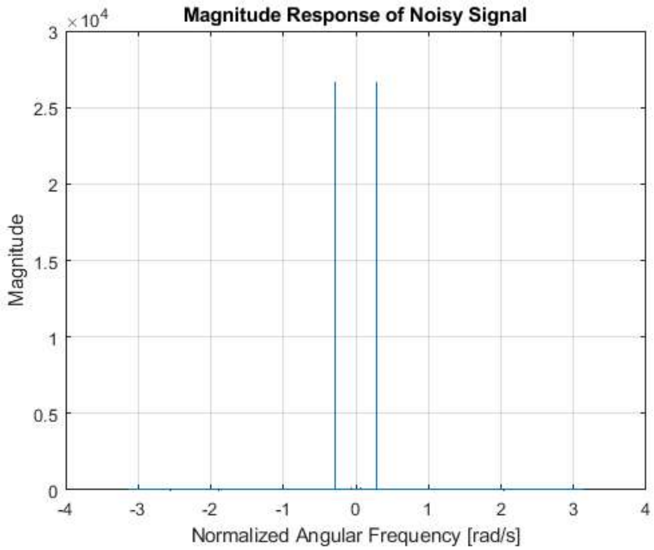
4(a) EVALUATE DTFT OF INPUT SIGNAL

```
w = -pi:pi/10000:pi;                                        % Define a finer frequency range
H_noisy = DTFT(x, w);
```

```
% Plotting the DTFT of the noisy signal
figure;
plot(w, abs(H_noisy));
title('Magnitude Response of Noisy Signal');
xlabel('Normalized Angular Frequency [rad/s]');
ylabel('Magnitude');
grid on;
```


Time Domain Signal y[n] for (e)
Amplitude
n
Magnitude Response for (e)
Magnitude
Normalized Angular Frequency [rad/s]
Phase Response for (e)
Phase [rad]
Normalized Angular Frequency [rad/s]


Time Domain Signa
Amplitude
n
Magnitude Response
Magnitude
Normalized Angular Frequ
Phase Response fo
Phase [rad]
Normalized Angular Frequ

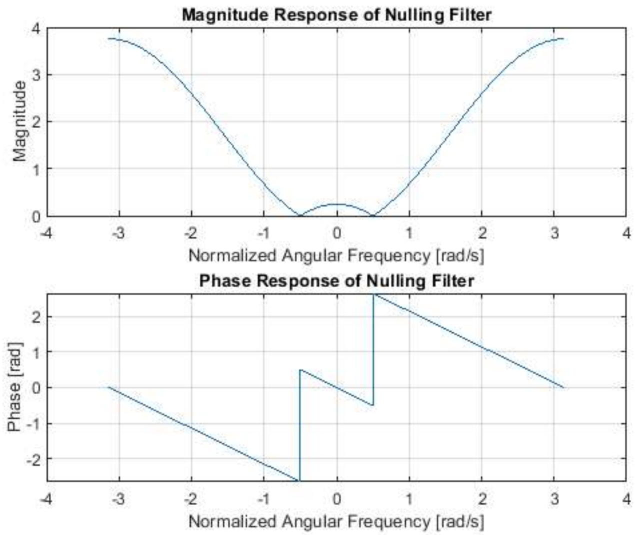


4(b) IDENTIFY FREQUENCY

```
wb0 = 0.5;                                                                          % Example normalized frequency
f_continuous = wb0 * fs / (2 * pi);                                                 % Continuous-time frequency

% <== ANSWER TO QUESTION ==>
%
% After analyzing the plot, identify the frequency contaminated with noise.
% For example, if the noise is at a certain peak, you can use the data tips in MATLAB to find the frequency.
% Let's say the noise is at normalized frequency wb0 = 0.5 (example).
% The continuous-time cyclic frequency can be calculated as: f = wb0 * fs / (2*pi).
%
% The contaminated frequency is wb0 in normalized angular frequency and f_continuous in Hz.
```

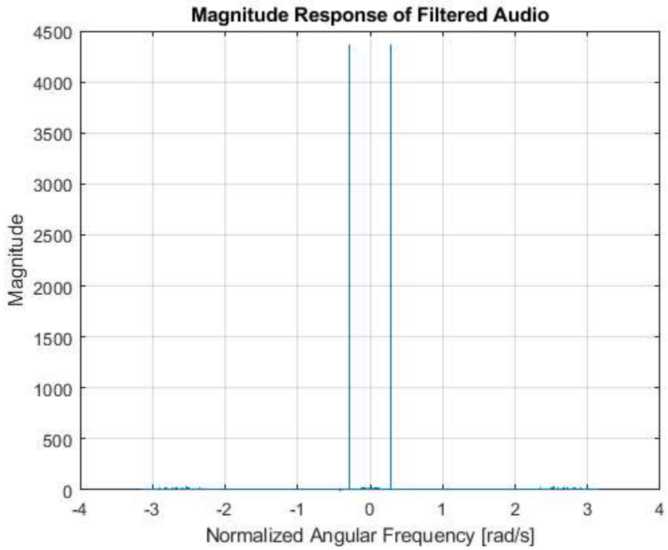
4(c) DESIGN FILTER

```
% Design the nulling FIR filter
wb0 = 0.5;                                                                          % Example normalized frequency
w = -pi:pi/10000:pi;                                                                % Define a finer frequency range
b = [1, -2*cos(wb0), 1];                                                            % Coefficients for the nulling filter
H_filter = DTFT(b, w);                                                              % DTFT of the filter

% Plotting the filter response
figure;
subplot(2,1,1);
plot(w, abs(H_filter));
title('Magnitude Response of Nulling Filter');
xlabel('Normalized Angular Frequency [rad/s]');
ylabel('Magnitude');
grid on;

subplot(2,1,2);
plot(w, angle(H_filter));
title('Phase Response of Nulling Filter');
xlabel('Normalized Angular Frequency [rad/s]');
ylabel('Phase [rad]');
grid on;
```



4(d) APPLY FILTER

```
% Apply the nulling filter to the noisy audio data
[x, fs] = audioread('noisy.wav');
filtered_audio = conv(x, b, 'same');                                        % Convolve the noisy signal with the filter

% Compute the DTFT of the filtered audio
H_filtered = DTFT(filtered_audio, w);

% Plotting the magnitude response of the filtered audio
figure;
plot(w, abs(H_filtered));
title('Magnitude Response of Filtered Audio');
xlabel('Normalized Angular Frequency [rad/s]');
ylabel('Magnitude');
grid on;

% <== ANSWER TO QUESTION ==>
%
% The differences between the original audio and the output of the nulling filter can be analyzed by comparing the magnitude responses.
% The filtered audio should have reduced noise at the contaminated frequency.
```



4(e) LISTEN TO AUDIO

```
% Save the filtered audio to a .wav file
audiowrite('filtered_audio.wav', filtered_audio, fs);

% Listen to the output of the nulling filter
sound(filtered_audio, fs);
```

4(f) EXTEND YOUR KNOWLEDGE TO NOISY2.WAV

```
% DO NOT REMOVE THE LINE BELOW
% MAKE SURE 'noisy2.wav' IS IN THE SAME DIRECTORY AS THIS FILE
[x, fs] = audioread('noisy2.wav');

% Repeat the process for noisy2.wav
% 4(a) EVALUATE DTFT OF INPUT SIGNAL
w2 = linspace(-pi, pi, 10000);                                  % Define a finer frequency range
H_noisy2 = DTFT(x, w2);

% Plotting the DTFT of the noisy2 signal
figure;
plot(w2, abs(H_noisy2));
title('Magnitude Response of Noisy2 Signal');
xlabel('Normalized Angular Frequency [rad/s]');
ylabel('Magnitude');
grid on;

% 4(b) IDENTIFY FREQUENCY
% Analyze the plot to identify contaminated frequencies
% For example, let's say we identify multiple frequencies wb0_1, wb0_2, etc.
% You would repeat the filter design and application for each identified frequency.

% Example frequencies (you would replace these with actual identified frequencies)
wb0_1 = 0.3;                                                    % Example normalized frequency 1
wb0_2 = 0.7;                                                    % Example normalized frequency 2

% Design filters for each frequency
b1 = [1, -2*cos(wb0_1), 1];                                     % First nulling filter
```

```
b2 = [1, -2*cos(wb0_2), 1];                                    % Second nulling filter

% Apply the filters
filtered_audio1 = conv(x, b1, 'same');                         % Apply first filter
filtered_audio2 = conv(x, b2, 'same');                         % Apply second filter

% Save the filtered audio to .wav files
audiowrite('filtered_audio1.wav', filtered_audio1, fs);
audiowrite('filtered_audio2.wav', filtered_audio2, fs);

% Listen to the output of the nulling filters
sound(filtered_audio1, fs);
pause(length(filtered_audio1)/fs + 1);                         % Wait for the first sound to finish
sound(filtered_audio2, fs);

% <== ANSWER TO QUESTION ==>
% The approach for noisy2.wav involves identifying multiple contaminated frequencies and applying the nulling filter for each frequency.
% The output should reveal the underlying audio with reduced noise.
```

ALL FUNCTIONS SUPPORTING THIS CODE

```
function H = DTFT(x,w)
%  ===>
% This function computes the Discrete-Time Fourier Transform (DTFT) of a discrete-time signal x
% for a given set of angular frequencies w. The DTFT is calculated using the formula:
% H(e^jw) = sum(x[n] * e^(-jwn)), where n is the index of the signal x.
% <===

    H = zeros(length(w),1);                                    % Initialize the DTFT result vector
    for nn = 1:length(x)                                       % Loop over each sample in the input signal
        H = H + x(nn).*exp(-1j*w.'*(nn-1));                    % Accumulate the DTFT contributions
    end

end
```

Published with MATLAB® R2024a