Contents

QUESTION 2

DO NOT REMOVE THE LINE BELOW MAKE SURE 'eel3135_lab11_comment.m' IS IN THE SAME DIRECTORY AS THIS FILE

```
clear; close all; clc;
type('eel3135_lab10_comment.m')
```

```
%% Question 1 Comment Code
clear all; clc; close all;

fs = 16000;
tt = 0:1/fs:0.01-1/fs;

w0 = 888*pi;
w1 = 3520*pi;
w2 = 14080*pi;

x = 0.5 + cos(w0*tt) + cos(w1*tt + pi/4) + cos(w2*tt + 2*pi/3);
```

```
% Array with different numbers of samples
N = [50,length(x),500,1000];

X1 = fft(x,N(1));
X2 = fft(x,N(2));
X3 = fft(x,N(3));
X4 = fft(x,N(4));

% Answer in your comments: How is the DFT (using the FFT algorithm)
% calculated when N is larger than the length of X?
% (Hint: read help FFT)
% The DFT is calculated by adding zeros (zero-padding) the end of the
% x, so that the length of x matches the length of N.

% Answer on your comments: How is the DFT (using the FFT algorithm)
% calculated  when N is smaller than the length of X?
% When N is smaller than the length of X, X is truncated to the first few
% samples, so that the length of N matches the length of X, so this can
% eliminate a few of the samples that could be important to the input.

% Answer in your comments: Why would you not want to always make N as large
% as possible?
% It could slow down the FFT, making it difficult to do larger calculations
% or more difficult calculations, so using a large N is just really
% impractical for the most part.

b1 = -ceil((N(1)-1)/2):floor((N(1)-1)/2);
b2 = -ceil((N(2)-1)/2):floor((N(2)-1)/2);
b3 = -ceil((N(3)-1)/2):floor((N(3)-1)/2);
b4 = -ceil((N(4)-1)/2):floor((N(4)-1)/2);

% Answer in your comments: What frequencies do the peaks represent?
% The peaks are at +/- 0.0555pi (0.176rad/s), +/- 0.22pi (0.691rad/s),
% and +/- 0.88pi (2.76rad/s) I just converted the peaks to their normalized
% frequencies wrt to the sampling frequency.

f1 = b1*fs/N(1);
f2 = b2*fs/N(2);
f3 = b3*fs/N(3);
f4 = b4*fs/N(4);

w1 = 2*pi/N(1)*b1;
w2 = 2*pi/N(2)*b2;
w3 = 2*pi/N(3)*b3;
w4 = 2*pi/N(4)*b4;

figure
subplot(221)
plot(b1,fftshift(abs(X1)))
hold on;
plot(b1,fftshift(abs(X1)), '.', 'markersize', 8)
hold off;
xlim([-N(1)/2 N(1)/2]); ylim([0 100]);
xlabel('Bin Number')
ylabel('Magnitude')
title(sprintf('Length %i DFT',N(1)))
```

```
subplot(222)
plot(b2,fftshift(abs(X2)))
hold on;
plot(b2,fftshift(abs(X2)), '.', 'markersize', 8)
hold off;
xlim([-N(2)/2 N(2)/2]); ylim([0 100]);
xlabel('Bin Number')
ylabel('Magnitude')
title(sprintf('Length %i DFT',N(2)))

subplot(223)
plot(b3,fftshift(abs(X3)))
hold on;
plot(b3,fftshift(abs(X3)), '.', 'markersize', 8)
hold off;
xlim([-N(3)/2 N(3)/2]); ylim([0 100]);
xlabel('Bin Number')
ylabel('Magnitude')
title(sprintf('Length %i DFT',N(3)))

subplot(224)
plot(b4,fftshift(abs(X4)))
hold on;
plot(b4,fftshift(abs(X4)), '.', 'markersize', 8)
hold off;
xlim([-N(4)/2 N(4)/2]); ylim([0 81]);
xlabel('Bin Number')
ylabel('Magnitude')
title(sprintf('Length %i DFT',N(4)))



figure
subplot(221)
plot(f1,fftshift(abs(X1)))
hold on;
plot(f1,fftshift(abs(X1)), '.', 'markersize', 8)
hold off;
xlim([-fs/2 fs/2]); ylim([0 81]);
xlabel('Frequency (Hz)')
ylabel('Magnitude')
title(sprintf('Length %i DFT',N(1)))

subplot(222)
plot(f2,fftshift(abs(X2)))
hold on;
plot(f2,fftshift(abs(X2)), '.', 'markersize', 8)
hold off;
xlim([-fs/2 fs/2]); ylim([0 81]);
xlabel('Frequency (Hz)')
ylabel('Magnitude')
title(sprintf('Length %i DFT',N(2)))

subplot(223)
plot(f3,fftshift(abs(X3)))
hold on;
plot(f3,fftshift(abs(X3)), '.', 'markersize', 8)
hold off;
```

```
xlim([-fs/2 fs/2]); ylim([0 81]);
xlabel('Frequency (Hz)')
ylabel('Magnitude')
title(sprintf('Length %i DFT',N(3)))

subplot(224)
plot(f4,fftshift(abs(X4)))
hold on;
plot(f4,fftshift(abs(X4)), '.', 'markersize', 8)
hold off;
xlim([-fs/2 fs/2]); ylim([0 81]);
xlabel('Frequency (Hz)')
ylabel('Magnitude')
title(sprintf('Length %i DFT',N(4)))
```

```
figure
subplot(221)
plot(w1,fftshift(abs(X1)))
hold on;
plot(w1,fftshift(abs(X1)), '.', 'markersize', 8)
hold off;
xlim([-pi pi]); ylim([0 81]);
xlabel('Normalized Frequency (rad/s)');
ylabel('Magnitude')
title(sprintf('Length %i DFT',N(1)))

subplot(222)
plot(w2,fftshift(abs(X2)))
hold on;
plot(w2,fftshift(abs(X2)), '.', 'markersize', 8)
hold off;
xlim([-pi pi]); ylim([0 81]);
xlabel('Normalized Frequency (rad/s)');
ylabel('Magnitude')
title(sprintf('Length %i DFT',N(2)))

subplot(223)
plot(w3,fftshift(abs(X3)))
hold on;
plot(w3,fftshift(abs(X3)), '.', 'markersize', 8)
hold off;
xlim([-pi pi]); ylim([0 81]);
xlabel('Normalized Frequency (rad/s)');
ylabel('Magnitude')
title(sprintf('Length %i DFT',N(3)))

subplot(224)
plot(w4,fftshift(abs(X4)))
hold on;
plot(w4,fftshift(abs(X4)), '.', 'markersize', 8)
hold off;
xlim([-pi pi]); ylim([0 81]);
xlabel('Normalized Frequency (rad/s)')
ylabel('Magnitude')
```

```
title(sprintf('Length %i DFT',N(4)))

% Answer in your comments: How does the length of the DFT affect the
% magnitude? Be sure to zoom in on the bases of the peaks in each DFT
% magnitude plot.
% The magntiudes become more and more refined and larger for each peak.
% Therefore, at greater lengths of N, the peaks are actually closer to each
% other in terms of magnitude. Some of the peaks that were missed at much
% lower sample sizes are visible larger samples sizes as well (effect of
% truncating the input).
```

Question 2

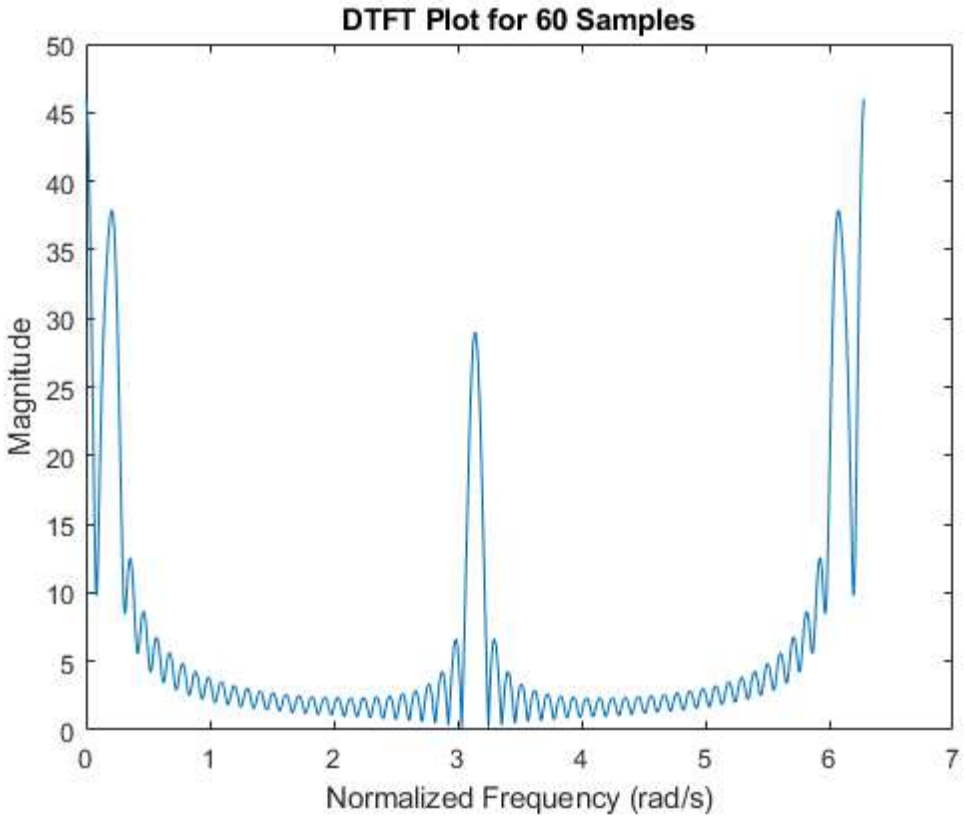
```
n=0:59;
x = 0.75 + cos(pi*n/20) + cos(pi*n/15) + cos(pi*n + 2*pi/3);
w_DTFT = linspace(0, 2*pi-pi/5000, 10000);

% NOTE: USE THE FOLLOWING COMMENTED LINE FOR PLOTTING THE DFT ATOP THE DTFT
% (YOU NEED TO DEFINE w_DFT), THIS WILL MAKE THE PLOTS EASIER TO INTERPRET
% plot(w_DTFT,abs(X_DTFT));
% hold on; plot(w_DFT,abs(X_DFT),'.', 'markersize', 10); hold off;
```

Question 2a

Done

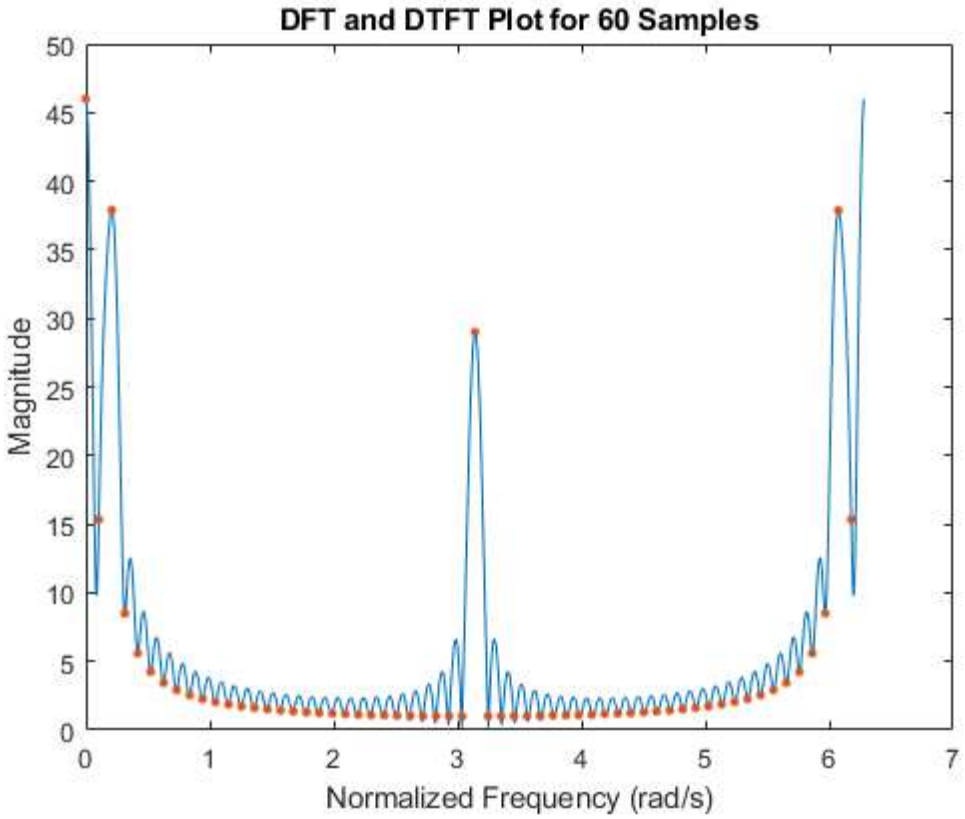
Question 2b

```
figure(1);
X_DTFT = DTFT(x, w_DTFT);
plot(w_DTFT, abs(X_DTFT));
title('DTFT Plot for 60 Samples');
xlabel('Normalized Frequency (rad/s)');
ylabel('Magnitude');
```



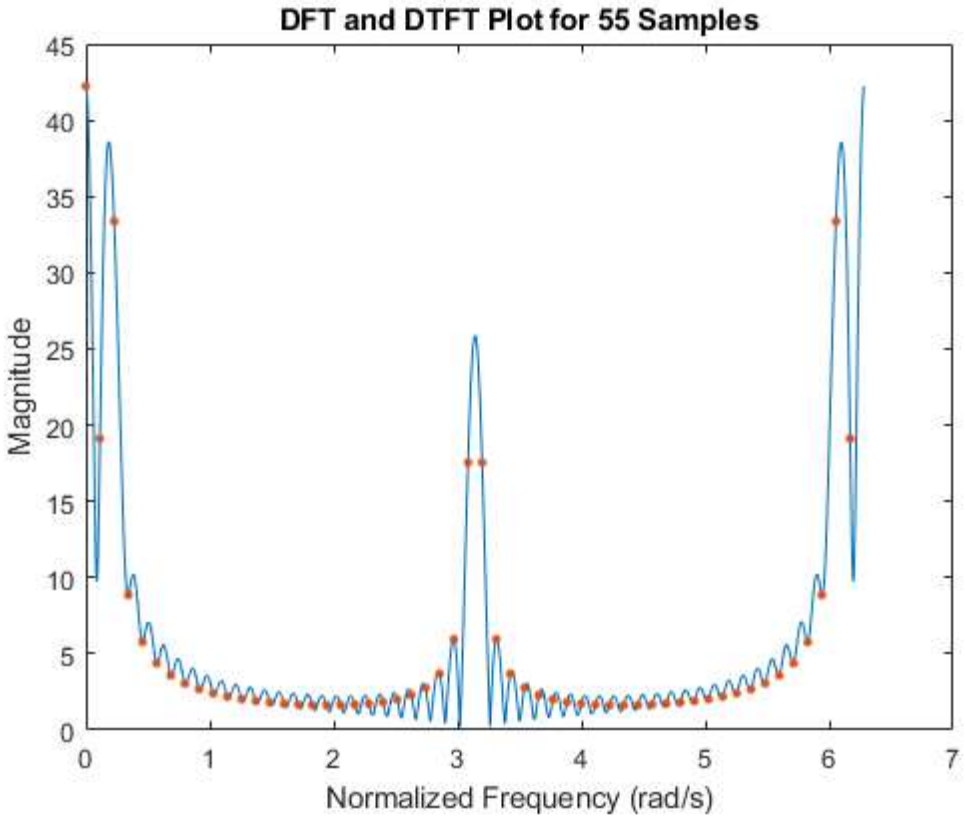
Question 2c

```
figure(2);
w_DFT = 2*pi*(0:length(x)-1)/length(x);
X_DTFT = DTFT(x, w_DTFT);
X_DFT = DFT(x);
plot(w_DTFT, abs(X_DTFT));
hold on; plot(w_DFT, abs(X_DFT), '.', 'MarkerSize',10); hold off;
title('DFT and DTFT Plot for 60 Samples');
xlabel('Normalized Frequency (rad/s)');
ylabel('Magnitude');
```



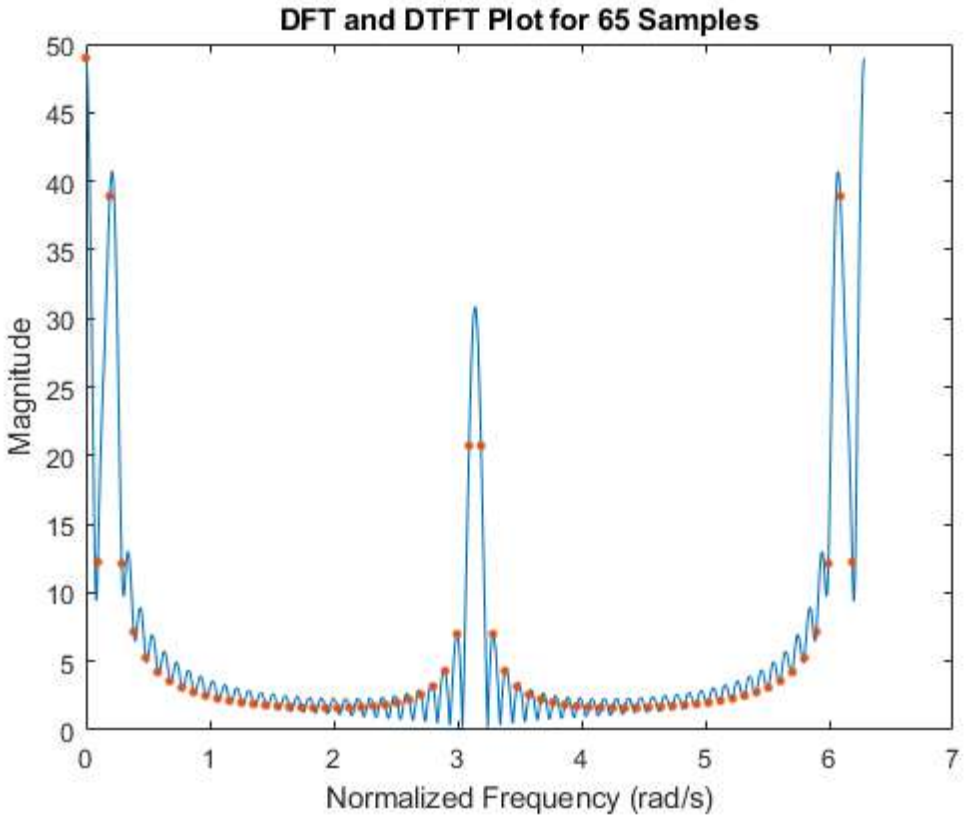
Question 2d

```
figure(3);
x = x(1:55);
w_DFT = 2*pi*(0:length(x)-1)/length(x);
X_DTFT = DTFT(x, w_DTFT);
X_DFT = DFT(x);
plot(w_DTFT, abs(X_DTFT));
hold on; plot(w_DFT, abs(X_DFT), '.', 'MarkerSize',10); hold off;
title('DFT and DTFT Plot for 55 Samples');
xlabel('Normalized Frequency (rad/s)');
ylabel('Magnitude');
```



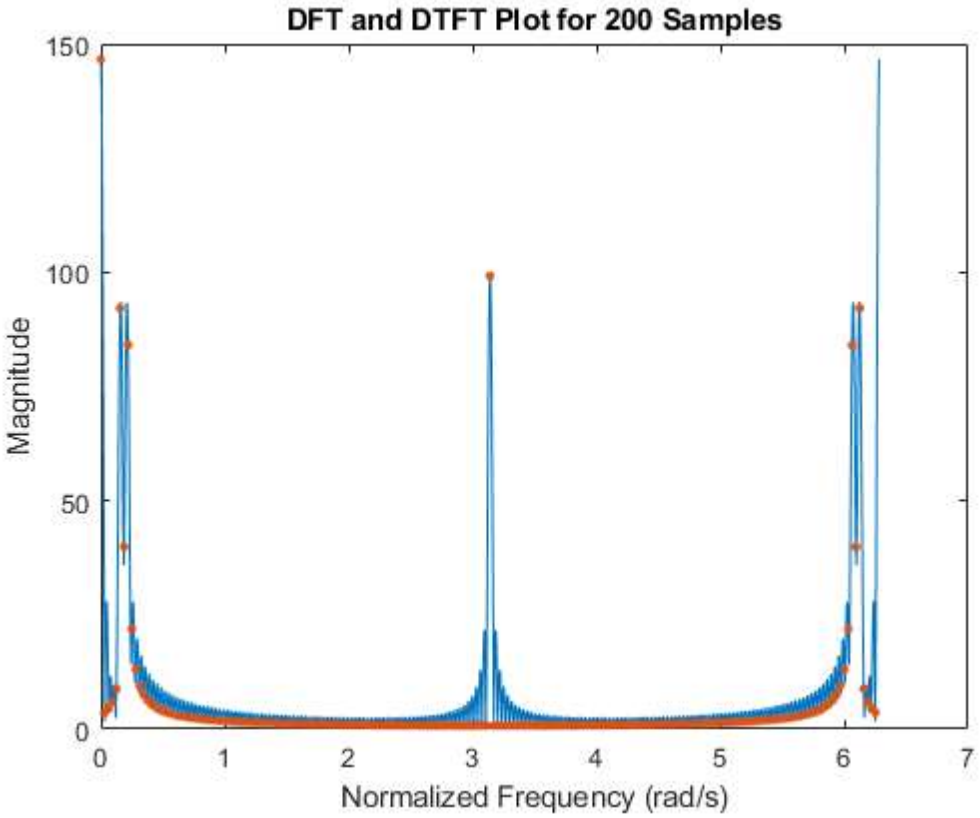
Question 2e

```
figure(4);
n = 0:64;
x = 0.75 + cos(pi*n/20) + cos(pi*n/15) + cos(pi*n + 2*pi/3);
w_DFT = 2*pi*(0:length(x)-1)/length(x);
X_DTFT = DTFT(x, w_DTFT);
X_DFT = DFT(x);
plot(w_DTFT, abs(X_DTFT));
hold on; plot(w_DFT, abs(X_DFT), '.', 'MarkerSize',10); hold off;
title('DFT and DTFT Plot for 65 Samples');
xlabel('Normalized Frequency (rad/s)');
ylabel('Magnitude');
```



Question 2f

```
figure(5);
n = 0:199;
x = 0.75 + cos(pi*n/20) + cos(pi*n/15) + cos(pi*n + 2*pi/3);
w_DFT = 2*pi*(0:length(x)-1)/length(x);
X_DTFT = DTFT(x, w_DTFT);
X_DFT = DFT(x);
plot(w_DTFT, abs(X_DTFT));
hold on; plot(w_DFT, abs(X_DFT), '.', 'MarkerSize',10); hold off;
title('DFT and DTFT Plot for 200 Samples');
xlabel('Normalized Frequency (rad/s)');
ylabel('Magnitude');
```



Question 2g

The DFT plot is effectively a DTFT plot but for the discrete-time domain. This is what it appears to be as we are effectively sampling a few points on the DTFT to get the DFT plot. This makes sense as instead of continuous omega_hat term, we use a discrete 2*pi*k/N term for our DFT transform. This is easier for computers to utilize as we are effectively sampling the DTFT with N samples. The result would be the same if we have over double the number of samples as the number of input values I assume (this could be like using the nyquist sampling frequency).

Question 3

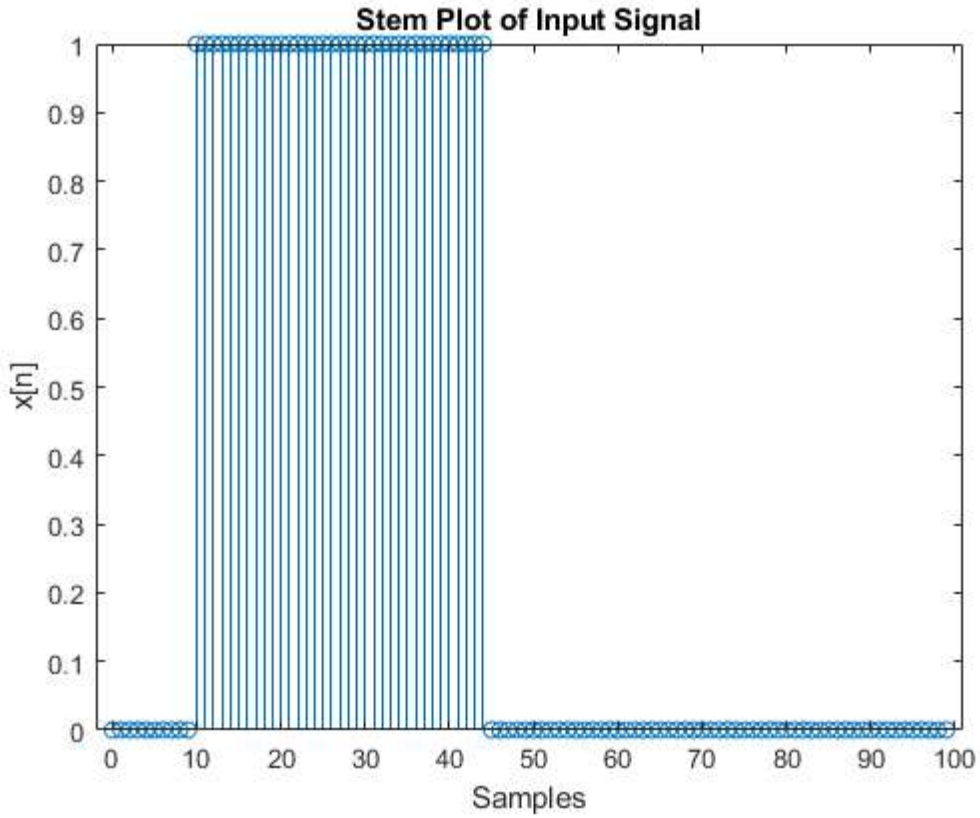
```
clear all; clc;
x = [zeros(1,10) ones(1,35)];
```

Question 3a

Done

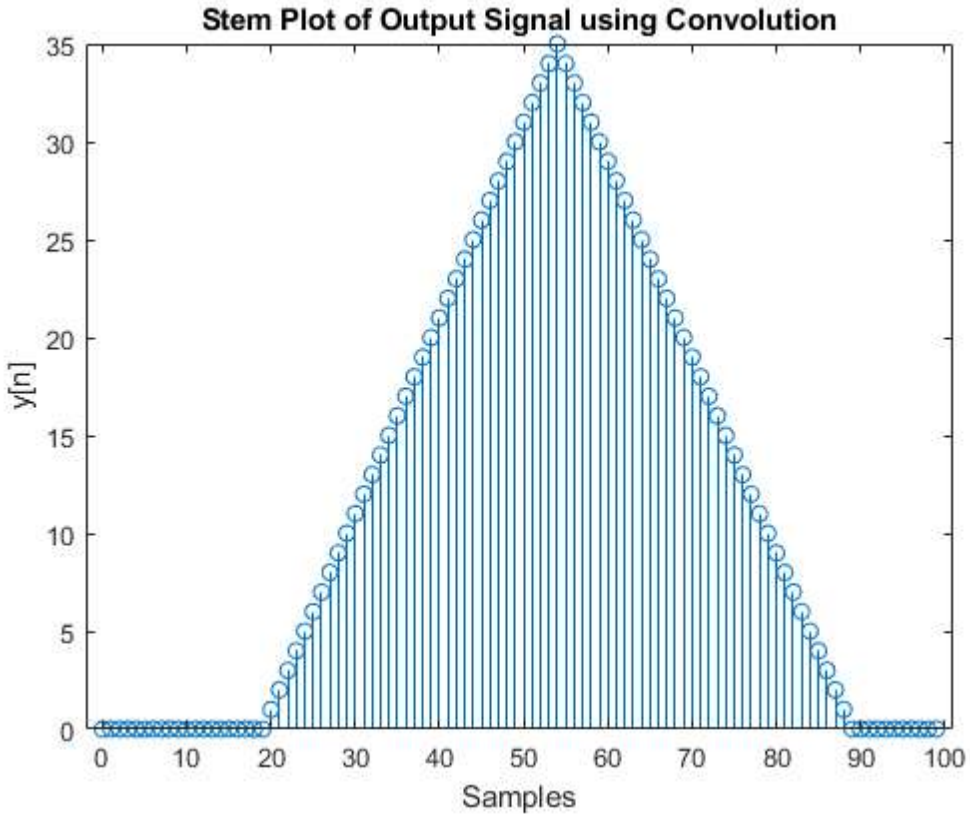
Question 3b

```
n = 0:99;
x_2 = [x zeros(1, 55)];
figure(6);
stem(n, x_2);
title('Stem Plot of Input Signal');
xlabel('Samples');
ylabel('x[n]');
```



Question 3c

```
y_n = conv(x_2, x_2);
y_sampled = y_n(1:100);
figure(7);
stem(n, y_sampled);
title('Stem Plot of Output Signal using Convolution');
xlabel('Samples');
ylabel('y[n]');
```

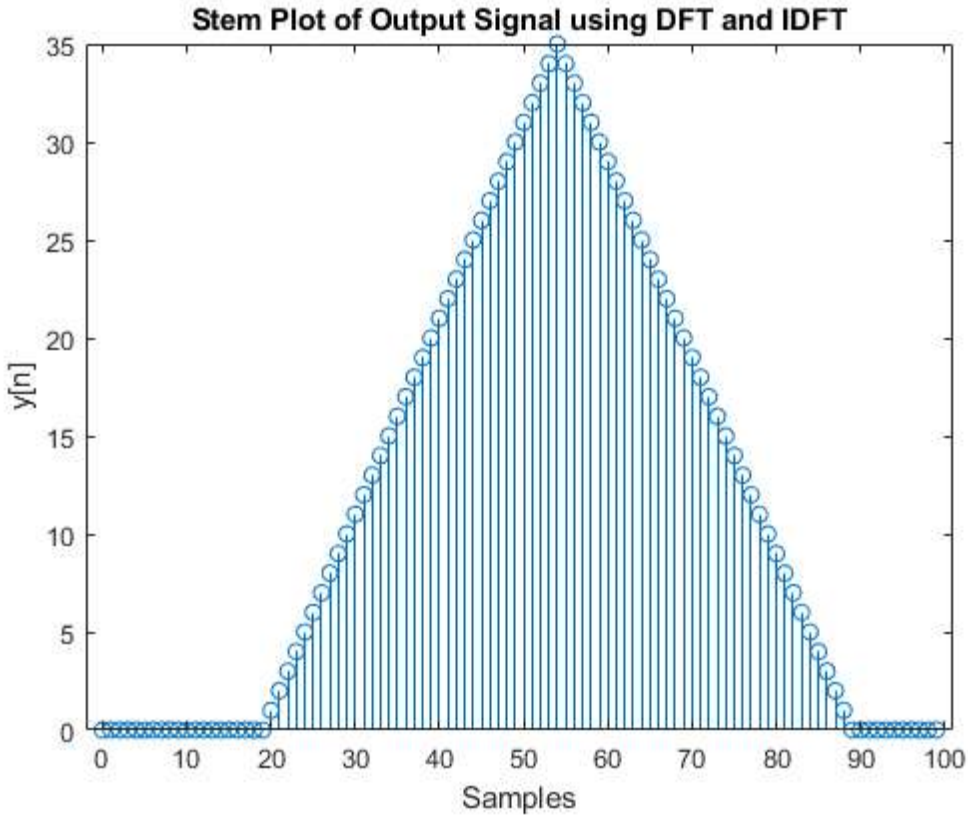


Question 3d

Step 1: Compute the DFT of X with length N = 100 used x_2 because it is length 100

```
X = DFT(x_2);

% Step 2: Multiply both DFTs together to get the frequency domain
% equivalent of convolution
X_s = X.*X;

% Step 3: Take the inverse DFT to get y_new
y_new = IDFT(X_s);

figure(8);
stem(n, y_new);
title('Stem Plot of Output Signal using DFT and IDFT');
xlabel('Samples');
ylabel('y[n]');
ylim([0, 35]);
```

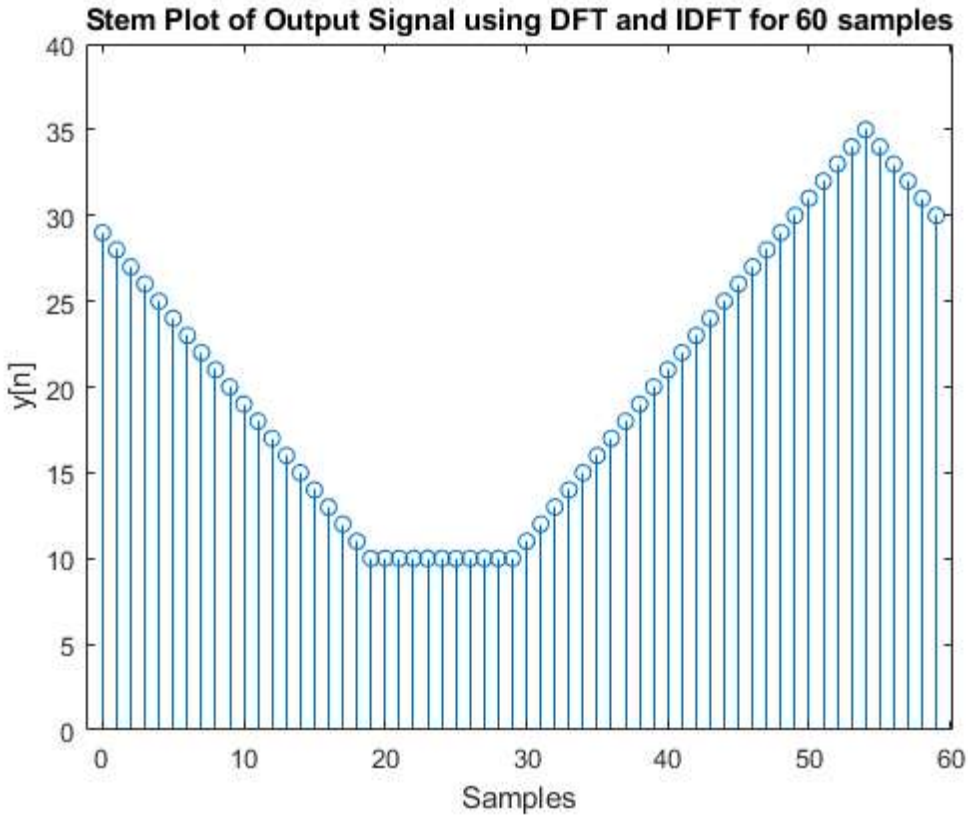
```
Warning: Using only the real component of complex data.
```



Question 3e

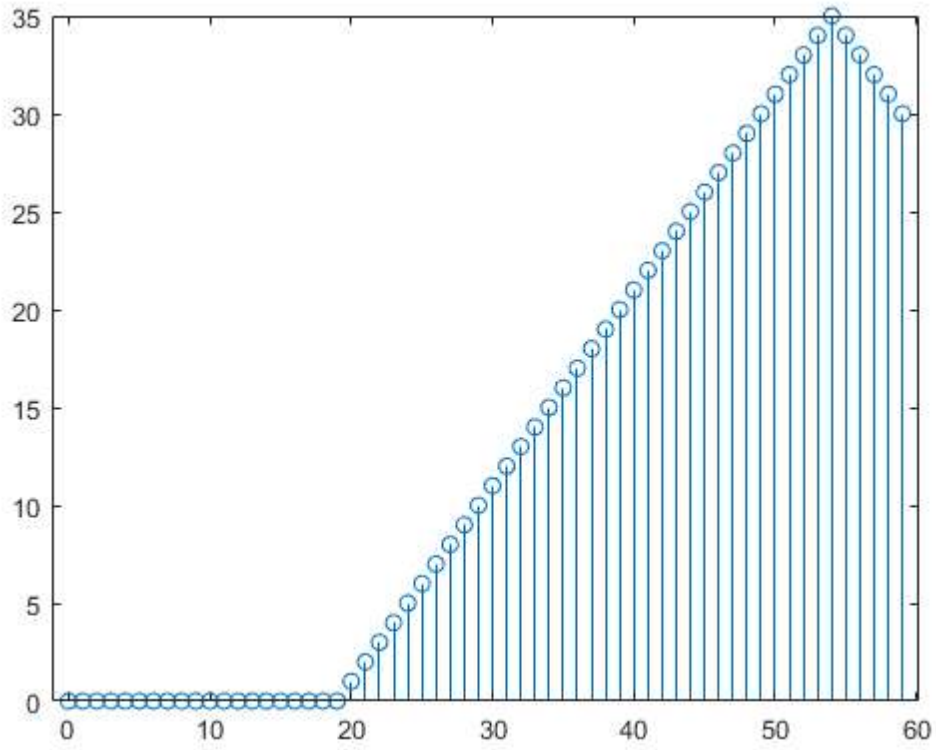
```
n2 = 0:59;
x_60 = [x zeros(1, 15)];

X_60 = DFT(x_60);

X_s60 = X_60 .* X_60;

y_60 = IDFT(X_s60);

figure(9);
stem(n2, y_60(1:60));
title('Stem Plot of Output Signal using DFT and IDFT for 60 samples');
xlabel('Samples');
ylabel('y[n]');
```

```
Warning: Using only the real component of complex data.
```



test

```
figure(10);
y_m = conv(x_60, x_60);
stem(n2, y_m(1:60));
```



Question 3f

There is a pretty large difference between the last two solutions which could be attributed to the zero-padding between them. This could be based on the fact that the DFT and IDFT rely on circular convolution, so if the length of the sampled signal is shorter than the length of the convolution, then some of the previous values wrap around and affect the DFT (aliasing). This happens with the length 60 input signal since it is not 2*length(x). However, the length 100 signal avoids this because it is padded to a value greater than 90, so will never actually wrap around.

Question 4

Choose a song at least 3 minutes long to use in this problem, and include it in your submission. Load it into MATLAB using audioread. Note that mose audio files will be stereo, so you need to make Sure that you only use one column of audio data for this part of the lab This site has a large archive of free music that you can choose from: https://freemusicarchive.org/static

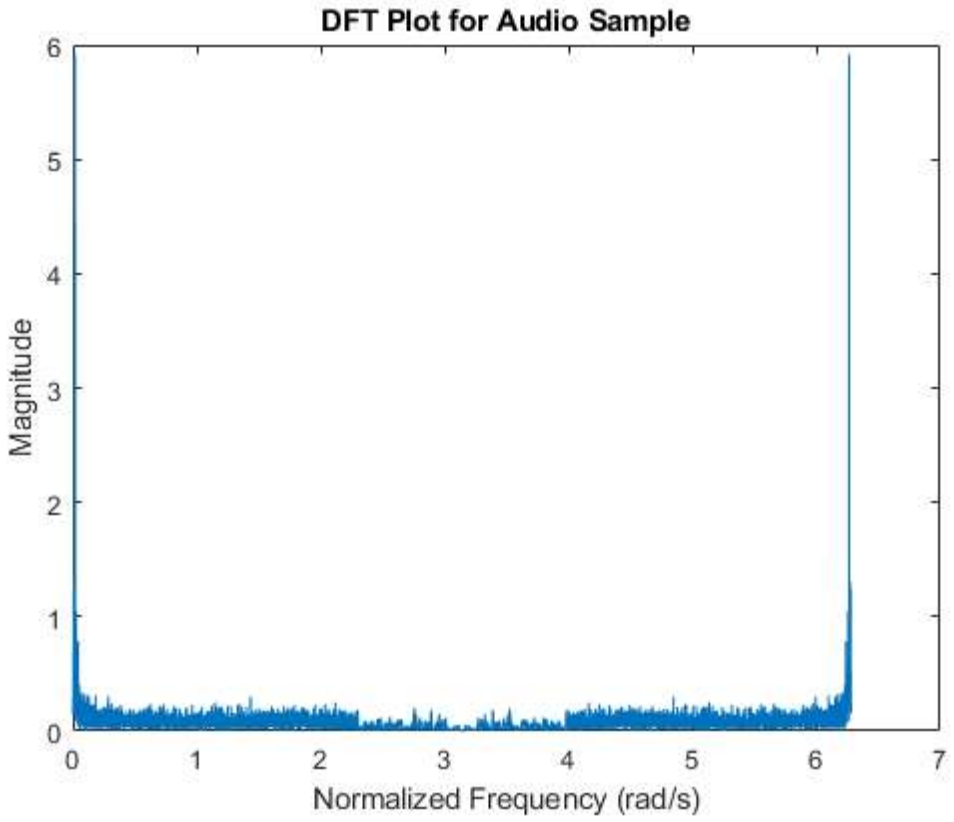
```
clear all; clc;
[x,fs] = audioread('Bruno Walter - Berlioz, Symphonie Fantastique 4th Mvt.mp3');
x = x(:,1);
```

Question 4a

```
x2 = x(1:10000);
n = 0:9999;
w_DFT = 2*pi*(0:length(x2)-1)/length(x2);

tic;
X = DFT(x2);
calcTime = toc;

% The calculation time was 1.1039 seconds when I tested it.
disp(['The calculation time is: ', num2str(calcTime), ' seconds']);
```

```
figure(11);
plot(w_DFT, abs(X));
title('DFT Plot for Audio Sample');
xlabel('Normalized Frequency (rad/s)');
ylabel('Magnitude');
```

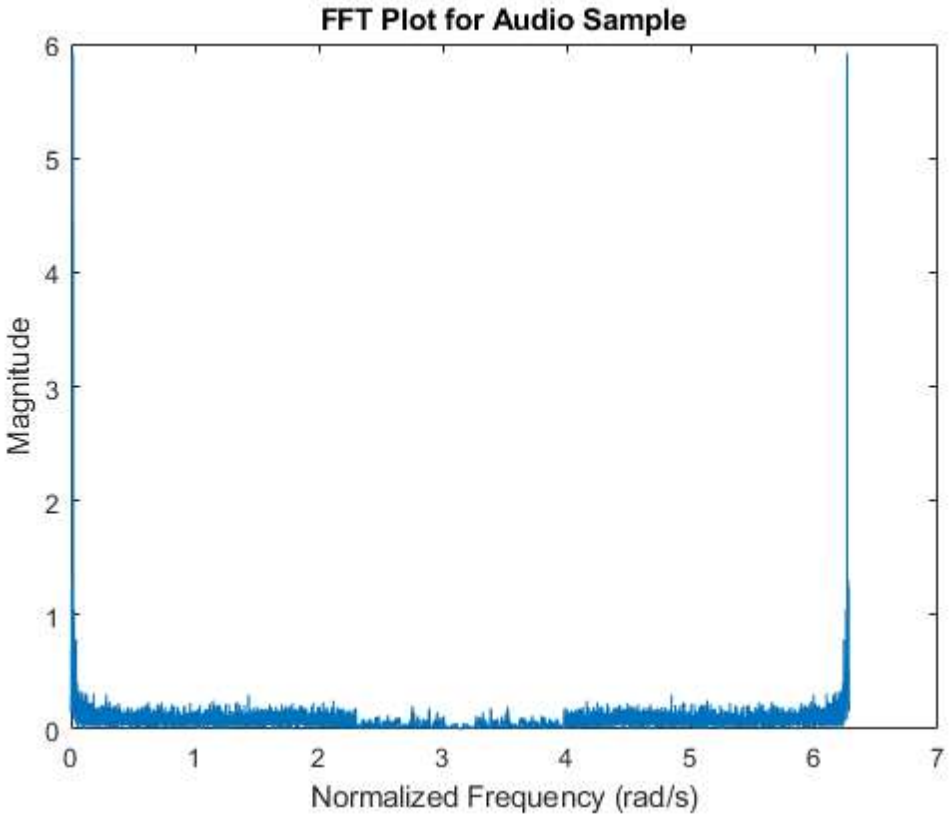
```
The calculation time is: 0.97001 seconds
```



Question

```
tic;
X = fft(x2, length(x2));
calcTime = toc;

% The calculation time was 0.0005234 seconds when I tested it.
disp(['The calculation time is: ', num2str(calcTime), ' seconds']);


figure(12);
plot(w_DFT, abs(X));
title('FFT Plot for Audio Sample');
xlabel('Normalized Frequency (rad/s)');
ylabel('Magnitude');
```

```
The calculation time is: 0.000255 seconds
```



Question 4c

There seem to be no differences in the magnitude results as far as I can tell. The audio sample is still relatively long with 10000 samples, so I never really looked to deeply into the differences, but both magnitude plots appear to be the same despite the difference in the way we calculated them.

Question 4d

The FFT is much, much faster than the DFT in this scenario. The DFT took approximately 1.1039 seconds to computer whereas the FFT only took 0.0005234 for the same calculation. This is a difference of 1.1033766 seconds which is an incredible difference given that the output was the same. Therefore, the FFT is just magnitudes faster in this case.

Question 4e

```
tic;
X = fft(x);
calcTime = toc;

% The calculation time was 0.20957 seconds when I tested it.
disp(['The calculation time is: ', num2str(calcTime), ' seconds']);
```

```
The calculation time is: 0.16489 seconds
```

Functions provided for the lab

```
function H = DTFT(x,w)
% DTFT(X,W)  compute the Discrete-time Fourier Transform of signal X
% acroess frequencies defined by W.
```

```
    H = zeros(1, length(w));
    for nn = 1:length(x)
        H = H + x(nn).*exp(-1j*w.*(nn-1));
    end

end

function X = DFT(x)
% DFT(x)  compute the N-point Discrete Fourier Transform of signal x
% Where N is the length of signal x
    N = length(x);
    w = 2*pi*(0:N-1)/N;
    X = zeros(1, length(w));
    for nn = 1:length(x)
        X = X + x(nn).*exp(-1j*w.*(nn-1));
    end
end

function x = IDFT(X)
% IDFT(x)  compute the N-point Inverse Discrete Fourier Transform of signal
% X where N is the length of signal X
    N = length(X);
    w = 2*pi*(0:N-1)/N;
    x = zeros(1, length(w));
    for nn = 1:length(x)
        x = x + X(nn)*exp(1j*w.*(nn-1));
    end
    x=x/N;
end
```

Published with MATLAB® R2024b