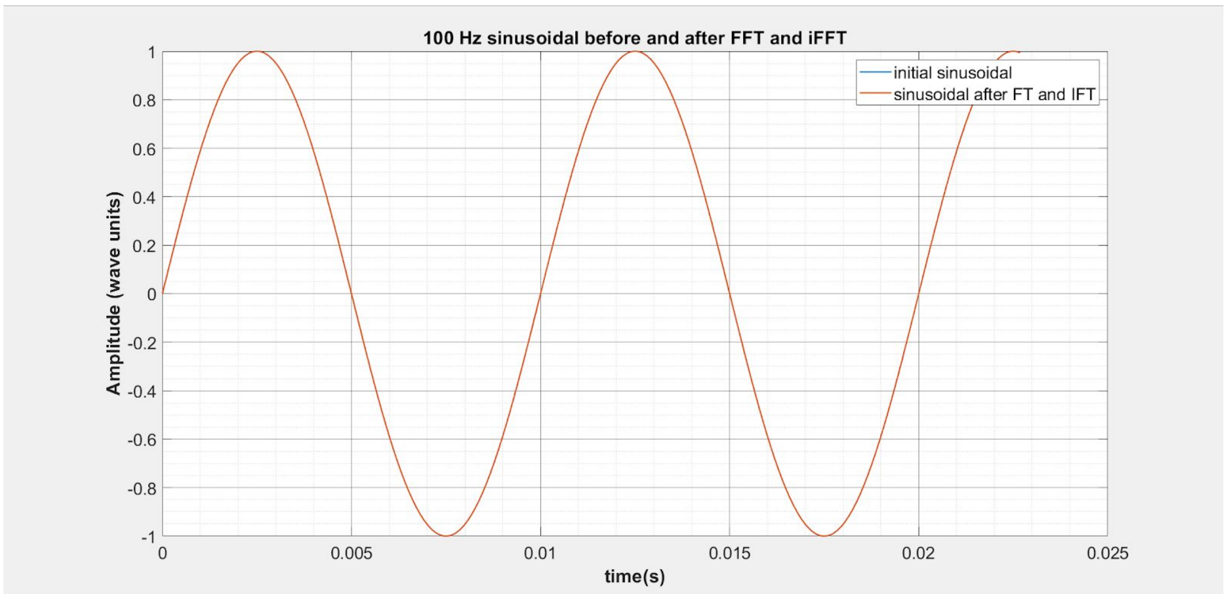
EID465 - HW1 - Yuval Epstain Ofek

My code with figures embedded (I used Code Blocks to make it more readable):

```
%%% Yuval Epstain Ofek
%%% Sound and Space - Problem Set 1

%% 1) Fixing MATLAB's fft function
% a. Create a new Matlab function that takes the FFT of a time series
%     vector and returns the correct value of the linear spectrum. Be
%     sure to shift the linear spectrum to the correct frequency vector.
% b. Similarly, create a new Matlab function that performs an IFFT on the
%     linear spectrum that you created in Part a.
% c. Demonstrate that these functions work by generating a sine wave
%     and plotting the time series and the results of your two new
%     functions. (Don't forget to perform a Parseval sum check!)
clear all;close all;clc;

% Parameters
N = 1001;
fs = 44100;

% Generating time series
t = (0:(N-1))/fs;
x = sin(2*pi*100*t);

% taking fft and ifft using the functions I made
[X, ~,check1, check2] = my_fft(x, N, fs);
% Displaying the values of each side of the Parseval's equality
check1
check2
[x2,icheck1, icheck2]= my_ifft(X,N,fs);
% Displaying the values of each side of the Parseval's equality
icheck1
icheck2
% We see that all 4 checks are the same, which tells us we probably took
% the FT properly.


% plotting the initial sinusoidal before the transforms and the output of
% the two transforms
figure
plot(t,x, t,x2, 'LineWidth', 1.25)
```

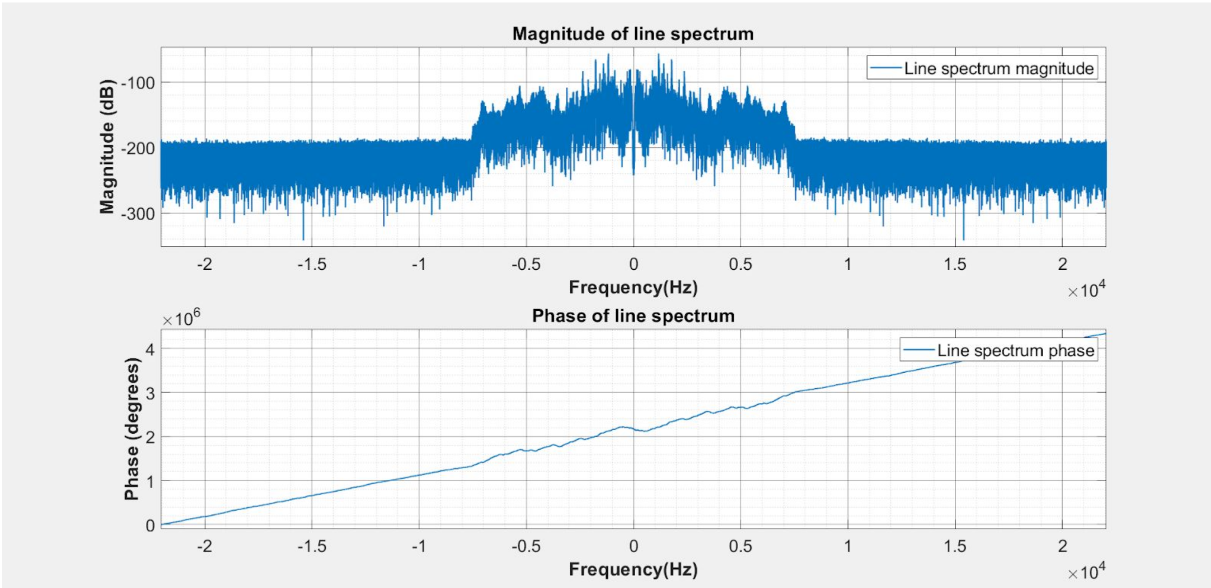
```
title('100 Hz sinusoidal before and after FFT and iFFT', 'FontSize', 18,
'FontWeight', 'bold')
xlabel('time(s)','FontSize', 16, 'FontWeight', 'bold')
ylabel('Amplitude (wave units)','FontSize', 16, 'FontWeight', 'bold')
grid on;
grid minor;
ax = gca;
ax.GridAlpha = 0.5;
ax.FontSize = 16;
legend('initial sinusoidal', 'sinusoidal after FT and IFT', 'FontSize', 16)
```




```
%% 2)  Recording and playback in Matlab
clear all;close all;clc;
% Recording time
T = 7;
fs = 44100;

% Setup to record
recording1 = audiorecorder(fs, 8, 1);
recording1.StartFcn = 'disp(''Start speaking.'')';
recording1.StopFcn = 'disp(''End of recording.'')';

%%% b) Record audio
recordblocking(recording1, T);

%%% c) Play what was recorded
```

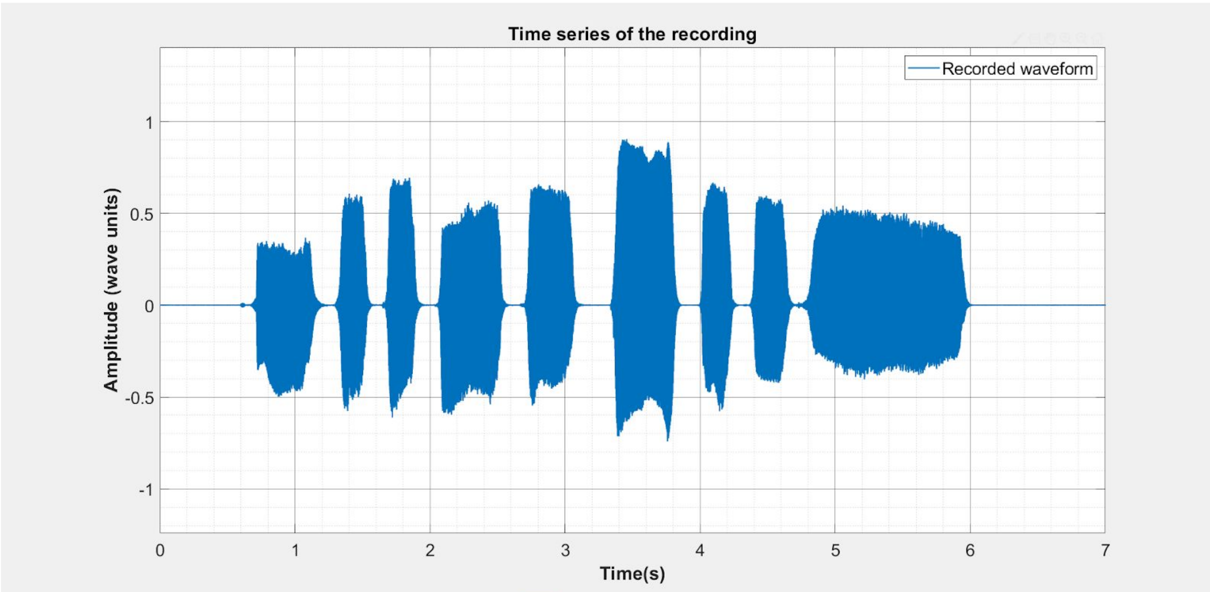
```
play(recording1);
%%% d) Plot time series
% Make recording data usable
myrecording = getaudiodata(recording1);

% Get the N used (can also find it from T and fs)
N = get(recording1,'TotalSamples');

% Create a time series
t = ((0:(N-1))/fs).';

% ploting the time series
figure
plot(t,myrecording, 'LineWidth', 1.25)
title('Time series of the recording', 'FontSize', 18, 'FontWeight', 'bold')
xlabel('Time(s)','FontSize', 16, 'FontWeight', 'bold')
ylabel('Amplitude (wave units)','FontSize', 16, 'FontWeight', 'bold')
grid on;
grid minor
ax = gca;
ax.GridAlpha = 0.5;
ax.FontSize = 16;
legend('Recorded waveform', 'FontSize', 16)
xlim([0,T])
ylim([min(myrecording)-0.5, max(myrecording)+0.5])

%%% e) Plot the magnitude and phase of the linear spectrum.
% Take FFT (while checking that we did it right with Parseval's)
[MYRECORDING,f, check1, check2]= my_fft(myrecording, N, fs);
% Displaying the values of each side of the Parseval's equality
check1
check2

% Computing Magnitude and Phase
Magrec = 20*log(abs(MYRECORDING));
Phaserec = unwrap(angle(MYRECORDING))*180/pi;

% Plotting magnitude
magbuff = 10;

figure
subplot(2,1,1)
```

```matlab
plot(f,Magrec, 'LineWidth', 1.25);
title('Magnitude of line spectrum', 'FontSize', 18, 'FontWeight', 'bold')
xlabel('Frequency(Hz)','FontSize', 16, 'FontWeight', 'bold')
ylabel('Magnitude (dB)','FontSize', 16, 'FontWeight', 'bold')
grid on;
grid minor
ax = gca;
ax.GridAlpha = 0.5;
ax.FontSize = 16;
legend('Line spectrum magnitude', 'FontSize', 16)
xlim([-fs/2, fs/2])
ylim([min(Magrec)-magbuff, max(Magrec)+magbuff])

% Plotting phase
phasebuff = 1e5;

subplot(2,1,2)
plot(f,Phaserec, 'LineWidth', 1.25);
title('Phase of line spectrum', 'FontSize', 18, 'FontWeight', 'bold')
xlabel('Frequency(Hz)','FontSize', 16, 'FontWeight', 'bold')
ylabel('Phase (degrees)','FontSize', 16, 'FontWeight', 'bold')
grid on;
grid minor
ax = gca;
ax.GridAlpha = 0.5;
ax.FontSize = 16;
legend('Line spectrum phase', 'FontSize', 16)
xlim([-fs/2, fs/2])
ylim([min(Phaserec)-phasebuff, max(Phaserec)+phasebuff])
```

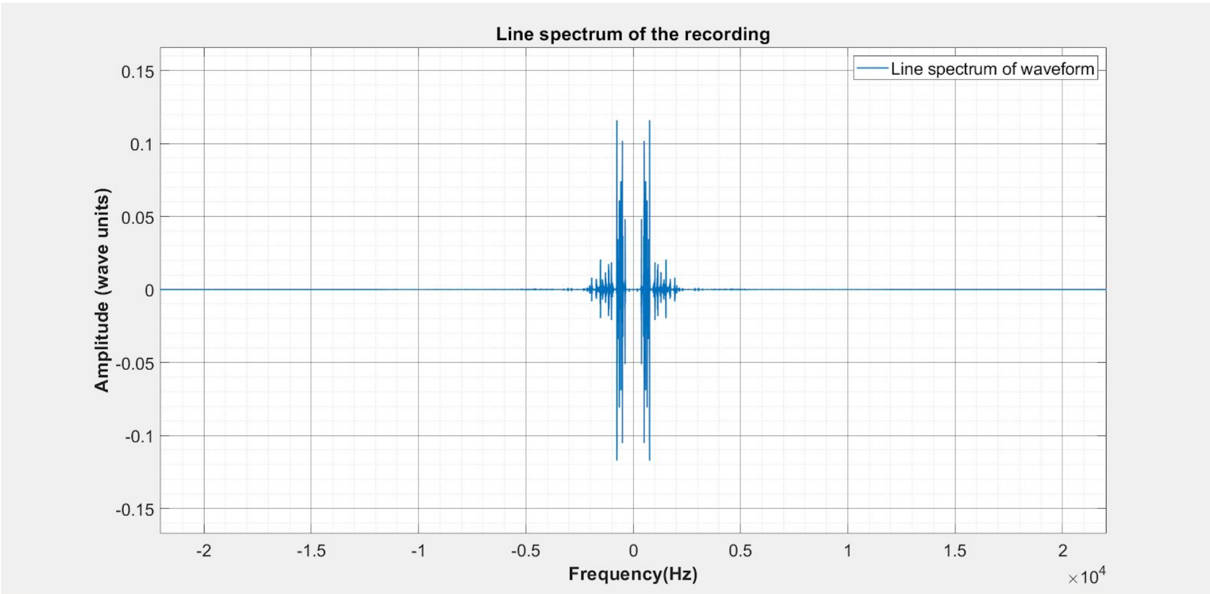



```
%%% f) save recording
filename = 'YuvalRecording.wav';
audiowrite(filename,myrecording,fs);

% Checking that I Saved properly:
% [y,Fs] = audioread(filename);
% sound(y,Fs)

%%
clear all;close all;clc;
[EEPhw1, fs] = audioread('EID 465 - HW 1 - EFP.wav');
N = size(EEPhw1, 1);
t = (0:(N-1))/fs;
T = N/fs;

%%% Ploting the time series
figure
plot(t,EEPhw1, 'LineWidth', 1.25)
title('Time series of the recording', 'FontSize', 18, 'FontWeight', 'bold')
xlabel('Time(s)','FontSize', 16, 'FontWeight', 'bold')
ylabel('Amplitude (wave units)','FontSize', 16, 'FontWeight', 'bold')
grid on;
grid minor
ax = gca;
ax.GridAlpha = 0.5;
ax.FontSize = 16;
```

```
legend('Recorded waveform', 'FontSize', 16)
xlim([0,T])
ylim([min(EEPhw1)-0.5, max(EEPhw1)+0.5])
```

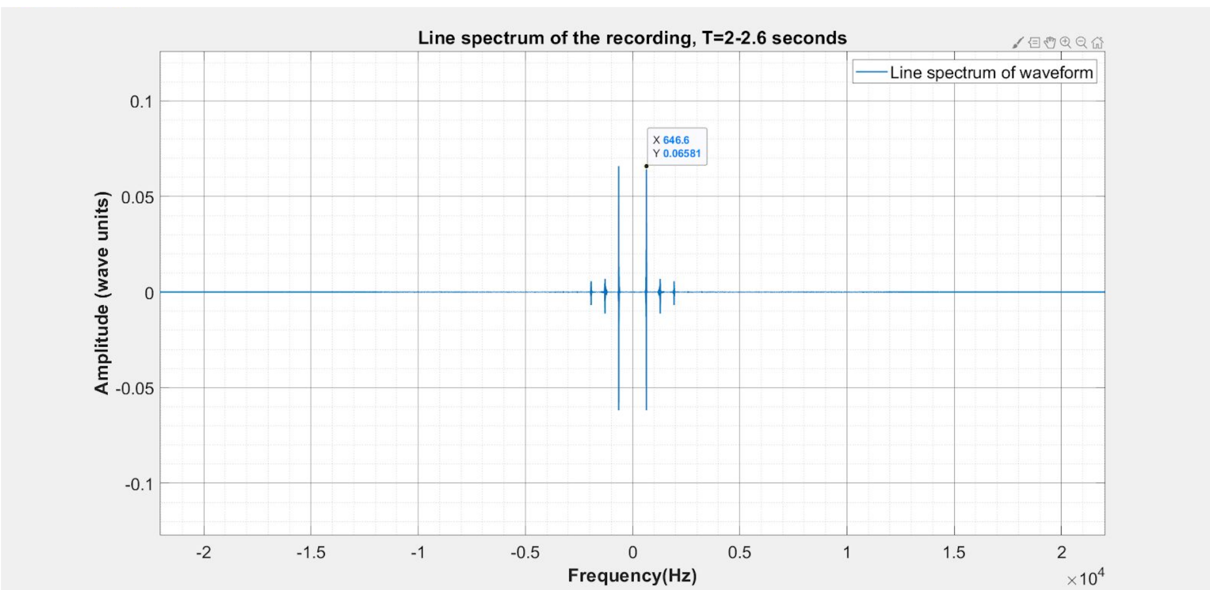



```
% Take FFT (while checking that we did it right with Parseval's)
[EEPHW1,f, check1, check2]= my_fft(EEPhw1, N, fs);
% Displaying the values of each side of the Parseval's equality
check1
check2

%%% Plotting the line spectrum
buff = 0.05;
figure
plot(f,EEPHW1, 'LineWidth', 1.25);
title('Line spectrum of the recording', 'FontSize', 18, 'FontWeight',
'bold')
xlabel('Frequency(Hz)','FontSize', 16, 'FontWeight', 'bold')
ylabel('Amplitude (wave units)','FontSize', 16, 'FontWeight', 'bold')
grid on;
grid minor
ax = gca;
ax.GridAlpha = 0.5;
ax.FontSize = 16;
legend('Line spectrum of waveform', 'FontSize', 16)
xlim([-fs/2, fs/2])
ylim([min(real(EEPHW1))-buff, max(real(EEPHW1))+buff])
```




```
% The time series plot tells us that during the 7 seconds of recording,
% there were 9 peaks of high amplitude, that were seperated by times that
% had low amplitude. Since our hearing is determined by the amplitude, we
% can say that there are 9 distinct sounds seperated by breaks with no
% sound. The line spectrum tells us the frequencies that were
% predominantly heard during the entire recording and their harmonics.
% We can dive deeper and extract each of the sounds (take the values at
% the time range the sound was heard) and then take individual FFTs to
% get the frequencies of the sounds that were heard.

%%% c) time extraction
% Provided time to extract
T1 = 2;
T2 = 2.6;
% Indices that correspond to these times
I1 = find(t == T1);
I2 = find(t == T2);
% Extracting the waveform in the specified time range
Extracted = EEPhw1(I1:I2);
% taking a FFT
[EXTRACTED,f] = my_fft(Extracted, I2-I1+1, fs);

% Check graphically what the predominant frequency is
buff = 0.01;
figure
```

```
plot(f,EXTRACTED, 'LineWidth', 1.25);
title('Line spectrum of the recording, T=2-2.6 seconds', 'FontSize', 18,
'FontWeight', 'bold')
xlabel('Frequency(Hz)','FontSize', 16, 'FontWeight', 'bold')
ylabel('Amplitude (wave units)','FontSize', 16, 'FontWeight', 'bold')
grid on;
grid minor
ax = gca;
ax.GridAlpha = 0.5;
ax.FontSize = 16;
legend('Line spectrum of waveform', 'FontSize', 16)
xlim([-fs/2, fs/2])
ylim([min(real(EEPHW1))-buff, max(real(EEPHW1))+buff])
```




```
% The predominant frequency is around 640 Hz

%% Functions:
%% my_fft
function [X,f,sum_check_t,sum_check_f] = my_fft(x,N, fs)
%[X,f,sum_check_t,sum_check_f,checkshift] = my_fft(x,N,fs)
%Takes the N point FFT of x and multiplies by 1/fs. Shifts the reference
%output and provides the shifted frequencies for plotting. Two outputs to
%check if the FT was performed properly, and one more to check if the shift
%is like fftshift.

%Make input row vector
```

```
x= x(:).';

dt = 1/fs;
df = fs/N;
%Take FT
Xpshift = fft(x,N)*dt;
%Checking if FT was good using Parseval's
sum_check_t= sum(x.^2)*dt;
sum_check_f= (sum(abs(Xpshift).^2))*df;

%shifting X
X = my_fftshift(Xpshift, N);

%Since my shift is like fftshift, the fs/2 component goes to -fs/2, which
%should be the first element of f:
f = [-floor(N/2):-1,0:ceil(N/2)-1]*df;
end

%% my_ifft
function [x, sum_check_t,sum_check_f] = my_ifft(X,N,fs)
%[x,sum_check_t,sum_check_f] = my_ifft(X,N,fs)
%Takes the N point iFFT of X after dividing X by dt. Two outputs to
%check if the FT was performed properly.

dt = 1/fs;
df = fs/N;
%Unshift - Note that this is NOT my_fftshift
X = [X(floor(N/2)+1:end), X(1:floor(N/2))];
%IFT
x = ifft(X)/dt;


%Checking if FT was good using Parseval's
sum_check_t= sum(x.^2)*dt;
sum_check_f= (sum(abs(X).^2))*df;
end

%% my_fftshift
function X = my_fftshift(X, N)
%X = myfftshift(X,N)
% My interpertation of fftshift. Takes the input and divides it into 2,
% when the size is odd, we let the make the first half (indices 1 to
```

```
% ceil(half)) be the bigger "half", and swap the two sections.
X = [X(ceil(N/2)+1:end), X(1:ceil(N/2))];
end
```