ECE366 Mini Project

Krishangakumar Senthilkumar

The screenshots in this report do not contain the comments which are in the code as the images would be difficult to fully capture with all the comments.

Problem 1

(a)

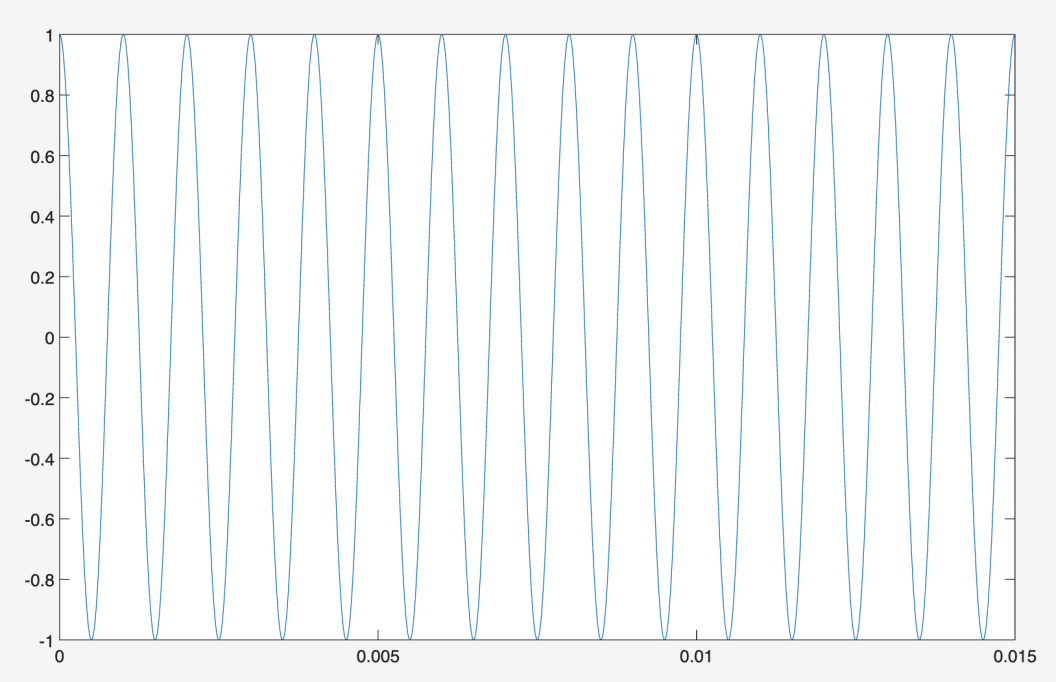

Figure 1: Plot for problem 1a

/Users/krish/Documents/Mini Project 2/ECE366_Mini_Project_2_Q1_Krishanga.m

```
1  % ECE366 Mini Project 2 Problem 1 - Krishanga
2  clear all; clc;
3
4  % Part a)
5  t = 0:1/96000:3;
6  w = 2*pi*1000;
7  t_plot = 0:1/96000:0.015;
8  plot(t_plot,cos(w*t_plot))
```

Figure 2: Code for problem 1a

(b) Figure 1 and figure 2 above shows the cosine wave for a 15ms timespan. I first created the timespan stated in problem 1 and created a separate span called t_plot to plot it for 15ms.

$$X(j\omega) = \int_{-\infty}^{\infty} \cos(2000\pi t) e^{-j2000\pi t} dt$$

Using the Fourier transform table we get:

$$= \pi(\delta(\omega - 2000\pi) + \delta(\omega + 2000\pi))$$

(c)

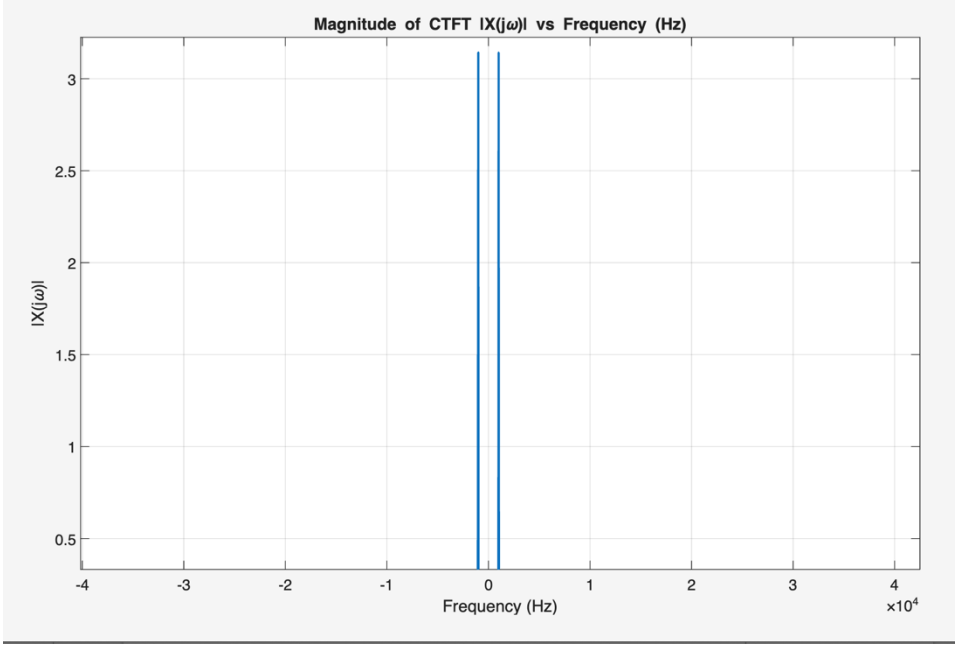

Figure 3: Plot for problem 1c

```
% Part c)
x = cos(w*t);
[X, w_axis, f_axis] = ctft(x, 96000);
figure;
plot(f_axis, abs(X));
grid on;
xlabel('Frequency (Hz)');
ylabel('|X(j\omega)|');
title('Magnitude of CTFT |X(j\omega)| vs Frequency (Hz)');
```

Figure 4: Plot for problem 1c

Figure 3 and figure 4 shows the code and the plot for the Fourier transform of x(t). From the plot we can see that the magnitude is 3.141..., which is the value of pi. This agrees with the derived Fourier transform from part b), where the 2 impulses have a magnitude of pi. The 2 impulses are also located at 1000 Hz and -1000Hz, which makes sense since this is the equivalent of being located at 2000pi and -2000pi if the angular frequency axis was used instead.

(d)

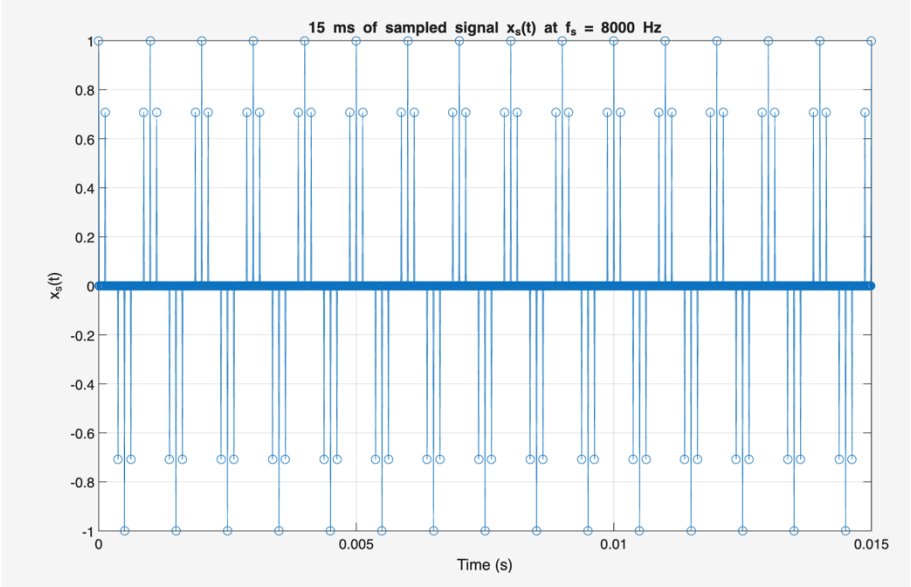

Figure 5: Plot for problem 1d

```
% Part d)
fs_samp = 8000;
Ts = 1/fs_samp;

s = zeros(size(t));
sample_n = round((0:Ts:3) * 96000) + 1;
sample_n(sample_n > length(t)) = [];
s(sample_n) = 1;

xs = x .* s;

figure;
plot(t_plot, xs(1:length(t_plot)), 'o-');
xlabel('Time (s)'); ylabel('x_s(t)');
title('15 ms of sampled signal x_s(t) at f_s = 8000 Hz');
grid on;
```

Figure 6: Code for problem 1d

Figure 5 and figure 6 shows the plot and code for the sampled signal. To create an approximation of the impulse train, the sample was created from 0 to 3 seconds, and this was multiplied by 96000 since this was a proposed fix posted on D2L. All the points that needed to be sampled were then added to a list of size t only containing zeros as given by the line s=zeros(size(t)) and were added to a list. The signal x (created in part a)) was then multiplied by the impulse train.

(e) From part b) we know that the Fourier transform of signal x(t) is given by:

$$X(j\omega) = \pi(\delta(\omega - 2000\pi) + \delta(\omega + 2000\pi))$$

Using the Fourier transform table we can also determine the transformation of impulse train to be,

$$S(j\omega) = \frac{2\pi}{T_s} \sum_{k=-\infty}^{\infty} \delta(\omega - k\omega_s)$$

Where T_s is the sampling period and ω_s is the sampling angular frequency. If we substitute the values we are provided, then we get:

$$S(j\omega) = 16000\pi \sum_{k=-\infty}^{\infty} \delta(\omega - 16000\pi k)$$

To then derive the Fourier transform of $X_s(j\omega)$, the convolution property can be utilized.

$$X_s(j\omega) = \frac{1}{2\pi} X(j\omega) * S(j\omega)$$

$$X_s(j\omega) = \frac{1}{2\pi} \pi(\delta(\omega - 2000\pi) + \delta(\omega + 2000\pi)) * 16000\pi \sum_{k=-\infty}^{\infty} \delta(\omega - 16000\pi k)$$

$$X_s(j\omega) = 8000\pi \sum_{k=-\infty}^{\infty} (\delta(\omega - 2000\pi - 16000\pi k) + \delta(\omega + 2000\pi - 16000\pi k))$$

(f)

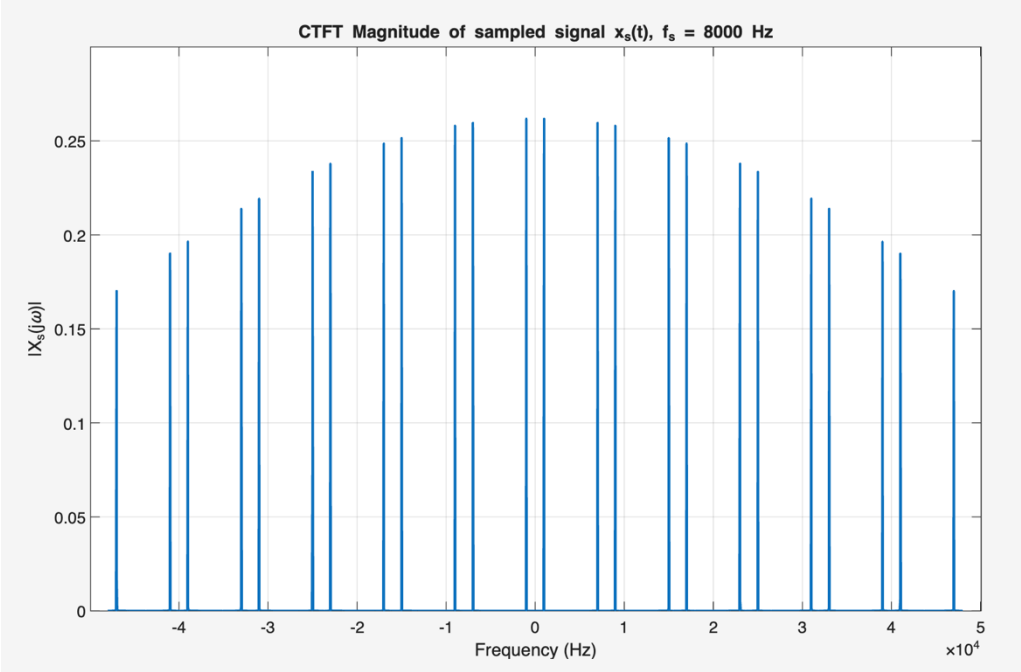

Figure 7: Plot for problem 1f

```
% Part f)
[Xs, w_axis2, f_axis2] = ctft(xs, 96000);

figure;
plot(f_axis2, abs(Xs));
xlabel('Frequency (Hz)'); ylabel('|X_s(j\omega)|');
title('CTFT Magnitude of sampled signal x_s(t), f_s = 8000 Hz');
grid on;
```

Figure 8: Code for problem 1f

From figure 7 we see the plot, with impulses at ±1000 Hz, ±7000Hz, ±9000Hz, etc. which agrees with our equation from part e). However, the difference lies in the magnitude of the impulse functions. While initially it appears to not agree with the equation derived the parabolic shape can

be explained by the fact that the Fourier transform was only calculated from 0 to 3s. This is essentially like a multiplying the signal with a sinc() function, which results in the parabolic shape. Instead had this Fourier transform been generated for all of infinity, then all the impulses would have the same magnitude. Thus, while it is not exactly what was derived, the plot is still accurate.

(g) To determine the cutoff frequency that we will use, we notice that from part f), our spectra contain the sampled points at ±1000 Hz, ±7000Hz, ±9000Hz, etc. Thus the ±1000 Hz frequencies contain all the information that we need. Since the next set of sampled points exists at ±7000Hz and beyond, any frequency between 1000Hz and 7000Hz would be optimal. For this case I will use a cutoff frequency of 4000Hz as this is right between 1000Hz and 7000Hz.

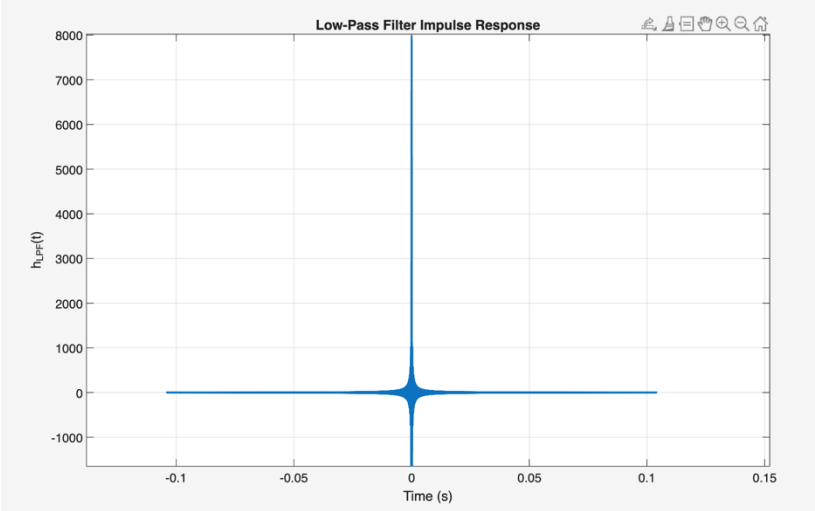

Figure 9: Plot for problem 1g

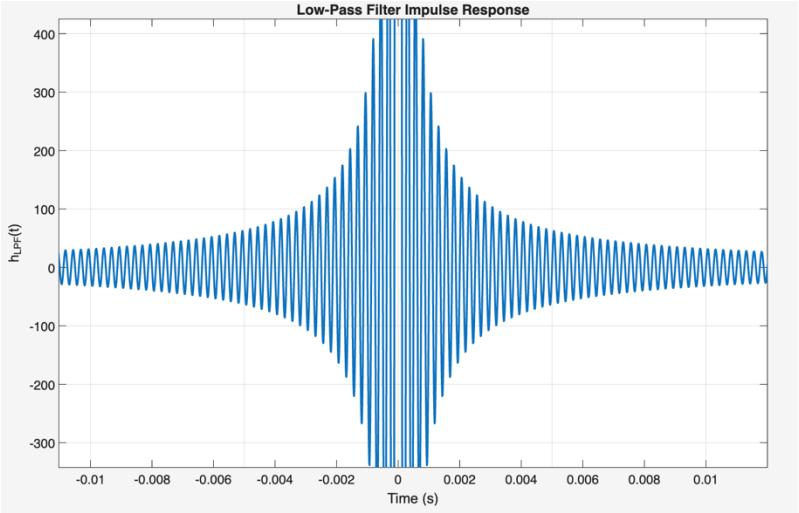

Figure 10: Plot for problem 1g

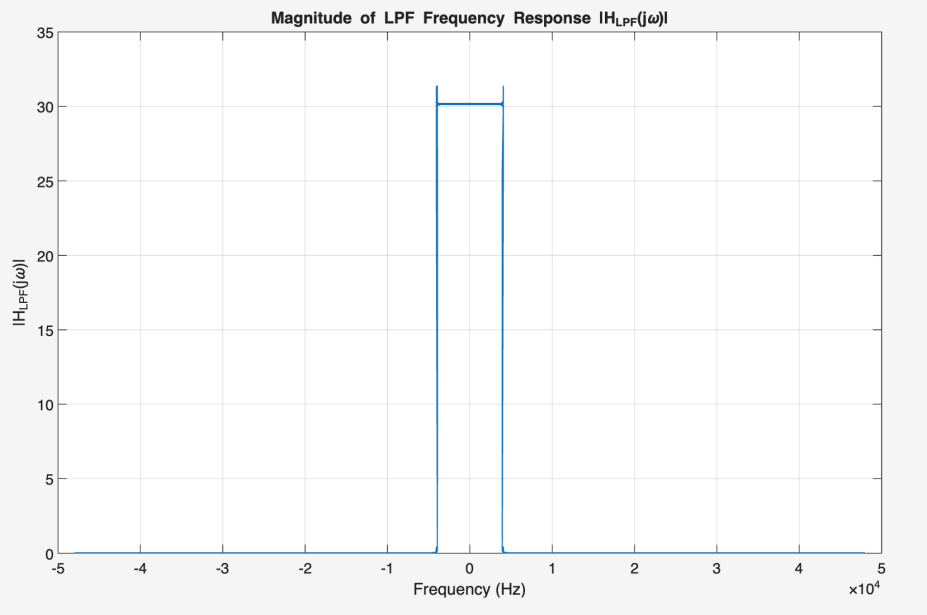

Figure 11: Plot for problem 1g

```
% Part g)
fc = 4000;

lpf_N = 20000;
lpft = linspace(-lpf_N/(2*96000), lpf_N/(2*96000), lpf_N);

hLPF = 2 * fc * sinc(2 * fc * lpft);

figure;
plot(lpft, hLPF);
xlabel('Time (s)'); ylabel('h_{LPF}(t)');
title('Low-Pass Filter Impulse Response');
grid on;

[HLPF, w_lpf, f_lpf] = ctft(hLPF, 96000);

figure;
plot(f_lpf, abs(HLPF));
xlabel('Frequency (Hz)'); ylabel('|H_{LPF}(j\omega)|');
title('Magnitude of LPF Frequency Response |H_{LPF}(j\omega)|');
grid on;
```

Figure 12: Code for problem 1g

Figures 9-11 show the graph of the low pass filter. Figure 10 was included as the frequency chosen resulted in a time period of 0.25ms, resulting in a plot that is hard to see unless zoomed in. However, from figure 9 and 10 we can observe that this is the expected shape of a sinc() function in the time domain and the time period from figure 10 matches the selected frequency of 4000Hz. In addition, we would expect the Fourier transform of this to look like a rect() function and figure 11 depicts this. Furthermore, we can observe the cutoff at 4000Hz. This matches the theoretical expectations. In addition, we see that the peak from figure 9 reaches 8000 units. This makes sense as our selected cutoff frequency is 4000 Hz, and at t=0, this filter will reach twice the cutoff frequency.

(h)

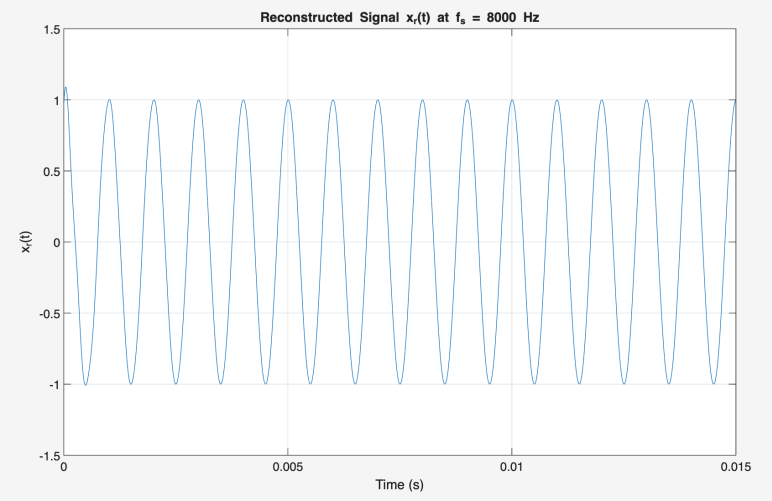

Figure 13: Plot for problem 1h

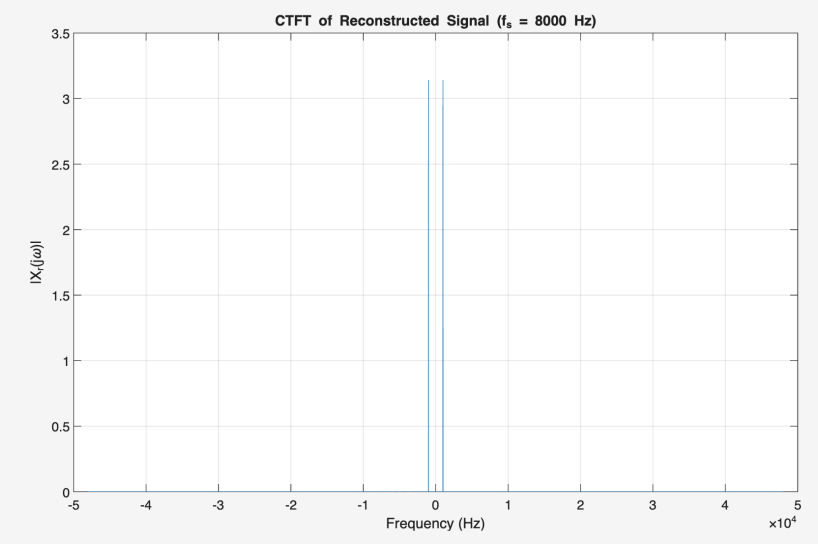

Figure 14: Plot for problem 1h

```
% Part h)
xr_full = conv(xs, hLPF);

mid = floor(length(hLPF)/2);
xr = xr_full(mid : mid + length(xs) - 1)* 1/8000;

figure;
plot(t_plot, xr(1:length(t_plot)));
xlabel('Time (s)');
ylabel('x_r(t)');
title('Reconstructed Signal x_r(t) at f_s = 8000 Hz');
grid on;

[Xr, w_r, f_r] = ctft(xr, 96000);

figure;
plot(f_r, abs(Xr));
xlabel('Frequency (Hz)');
ylabel('|X_r(j\omega)|');
title('CTFT of Reconstructed Signal (f_s = 8000 Hz)');
grid on;
```

Figure 15: Code for problem 1h

From figures 13 and 14 we see how the signal recovered is essentially identical to the initial signals created. We can also see the whole system behaving as intended as the time delay introduced from the LPF is also visible in figure 13. In addition, we also see the 2 identical impulses located at ±1000Hz of magnitude pi, which is a cosine wave in the time domain.

(i) Figures for f_s=3000Hz:
The two figures below show the LPF, and it agrees with theoretical calculations as there is a sinc() function with a peak of 3000 units, and the Fourier transform is a rect() function from -1500Hz to 1500Hz.

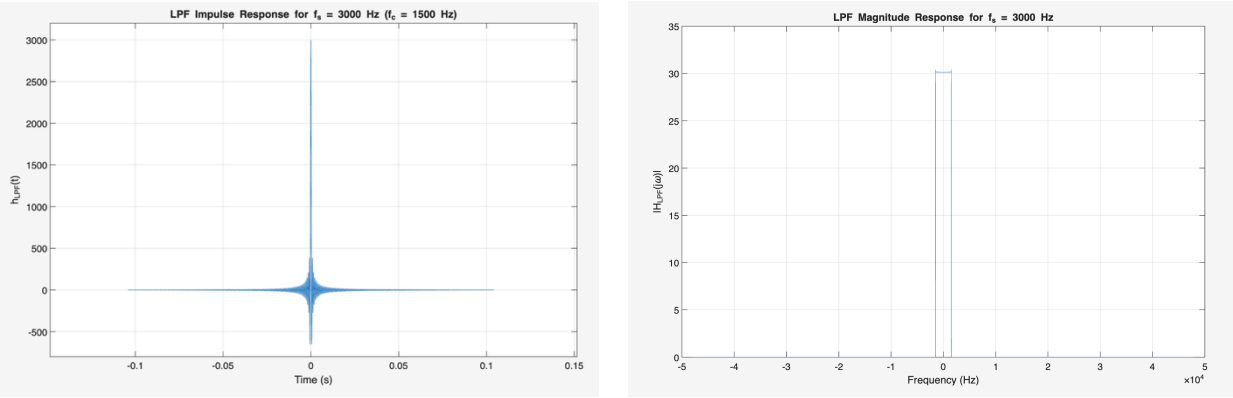

Figure 15: Plot for problem 1i

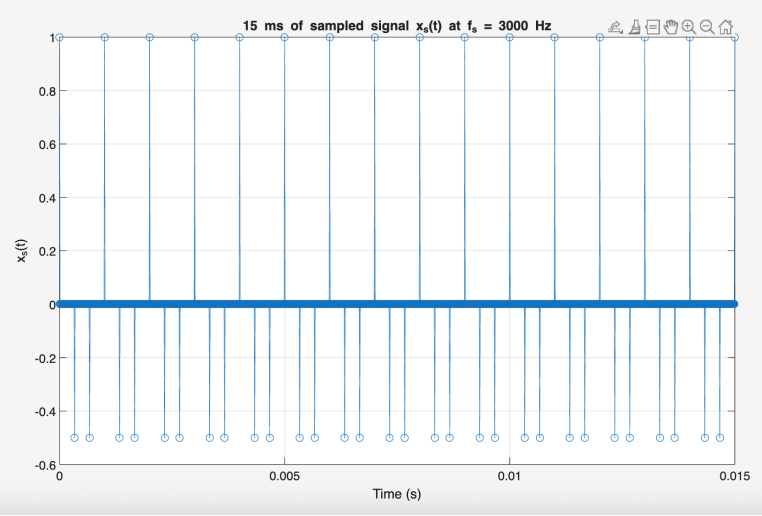

Figure 16: Plot for problem 1i

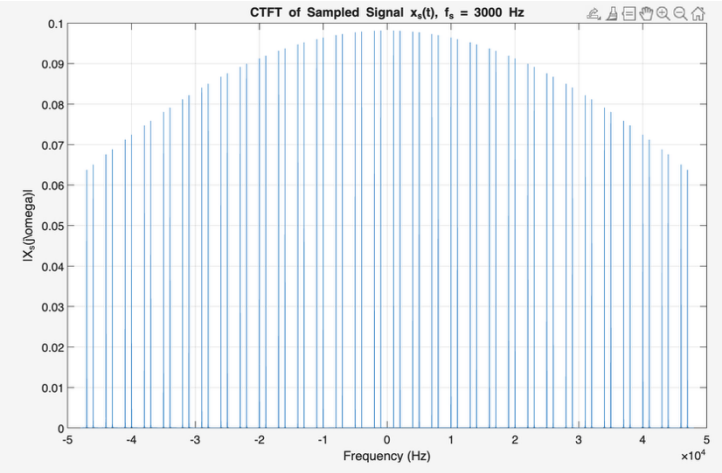

Figure 17: Plot for problem 1i

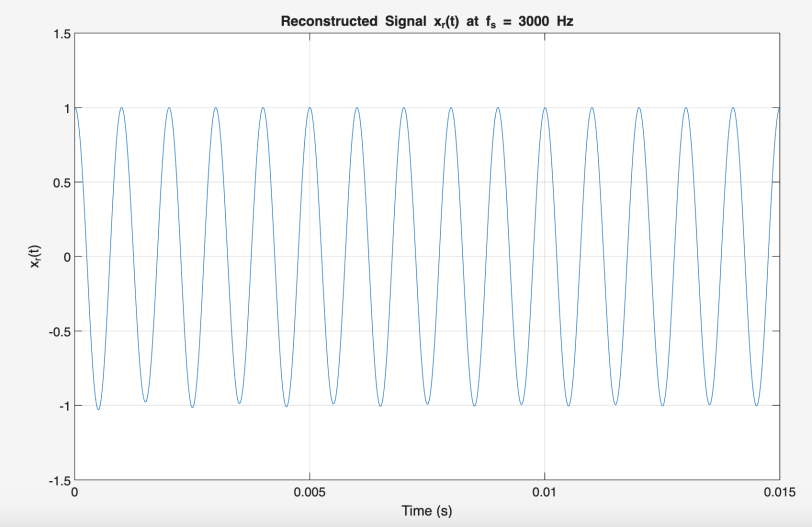

Figure 18: Plot for problem 1i

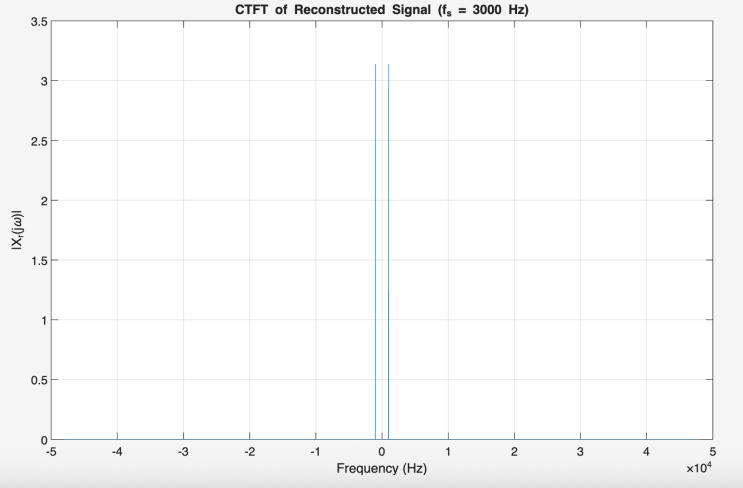

Figure 19: Plot for problem 1i

Using the same derivation as earlier if we substitute our value into our derived equation we get:

$$X_s(j\omega) = 3000\pi \sum_{k=-\infty}^{\infty} (\delta(\omega - 2000\pi - 6000\pi k) + \delta(\omega + 2000\pi - 6000\pi k))$$

From the figures above we see that since the sampling frequency of 3000Hz is above the Nyquist rate, which is 2000Hz, the recovered signal is a perfect reconstruction. This is expected as the sampling rate is high enough to perfectly map the continuous signal. If we observe the derived equation, we notice that this is identical to figure 17, indicating that the sampled signal is indeed correct and the simulated code matches theoretical expectations.

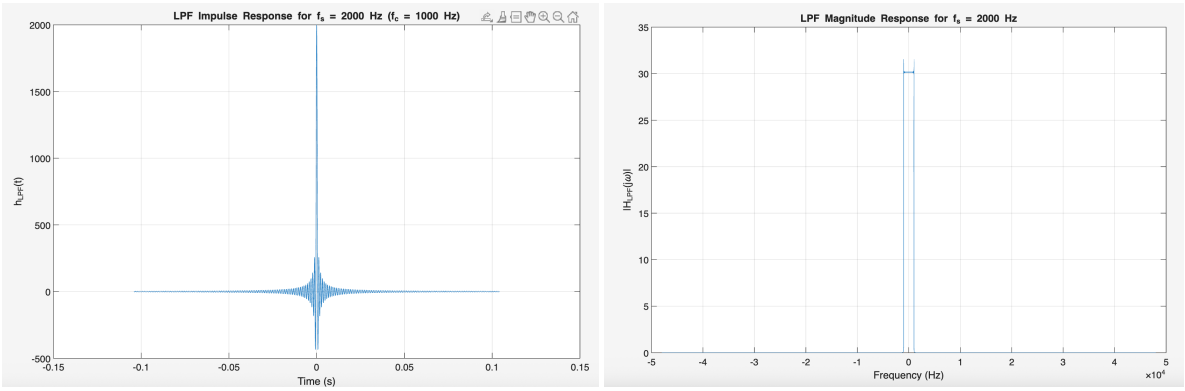

Figure 20: Plot for problem 1i

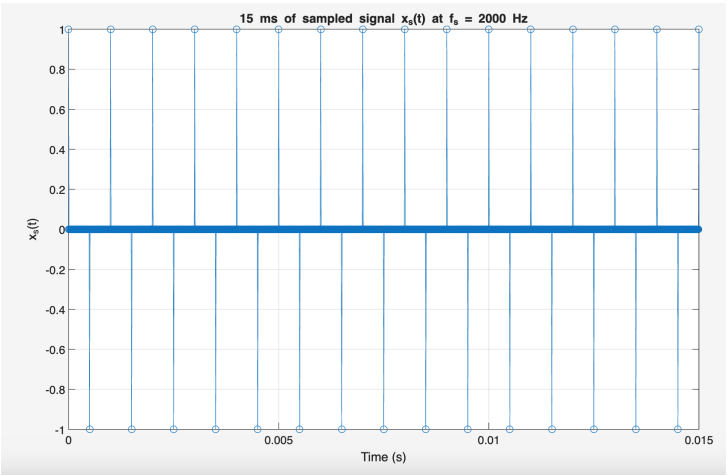

Figure 21: Plot for problem 1i

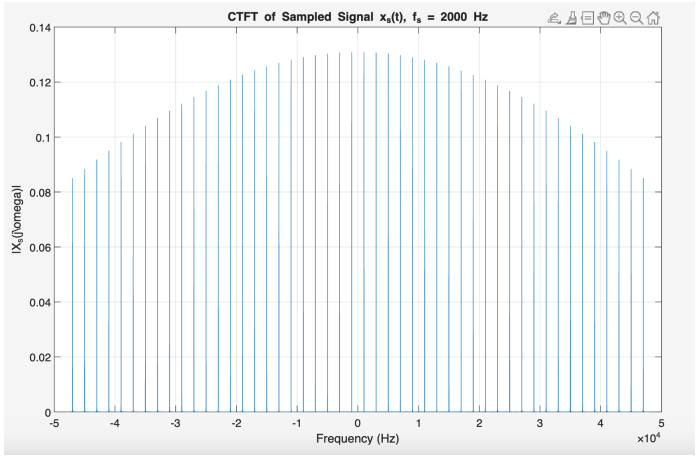

Figure 22: Plot for problem 1i

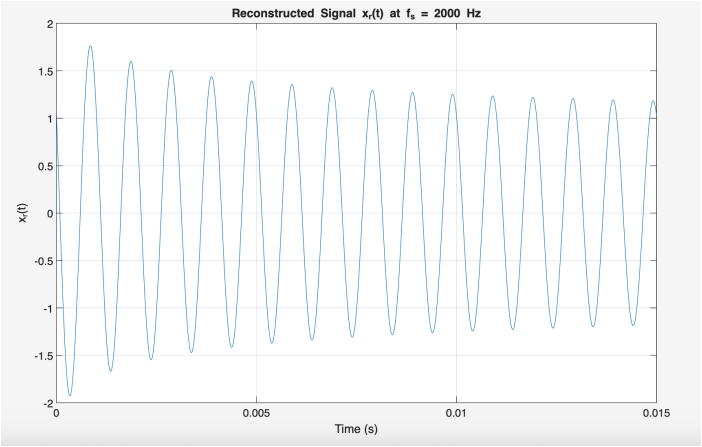

Figure 23: Plot for problem 1i

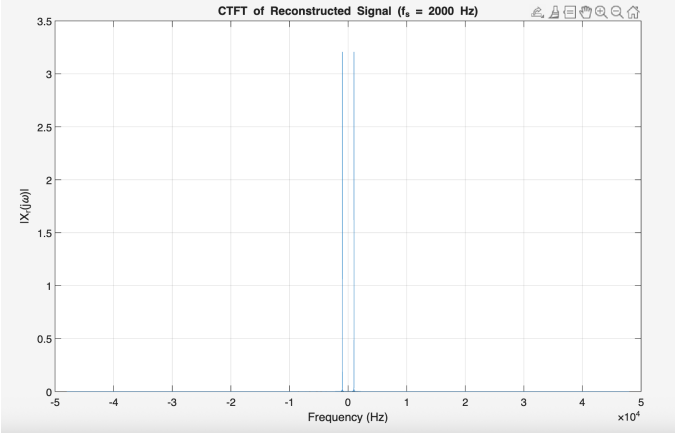

Figure 24: Plot for problem 1i

Using the same derivation as earlier if we substitute our value into our derived equation we get:

$$X_s(j\omega) = 2000\pi \sum_{k=-\infty}^{\infty} (\delta(\omega - 2000\pi - 4000\pi k) + \delta(\omega + 2000\pi - 4000\pi k))$$

From the figures above we see that since the sampling frequency of 2000Hz is exactly at the Nyquist rate, which is 2000Hz, the recovered signal is almost a perfect reconstruction. The distortions occur at the early time interval. This is expected as the sampling rate is at the edge of the Nyquist rate and perfectly maps the peaks of the x(t) signal as seen in figure 20. If we observe the derived equation, we notice that this is identical to figure 22, indicating that the sampled signal is indeed correct and the simulated code matches theoretical expectations.

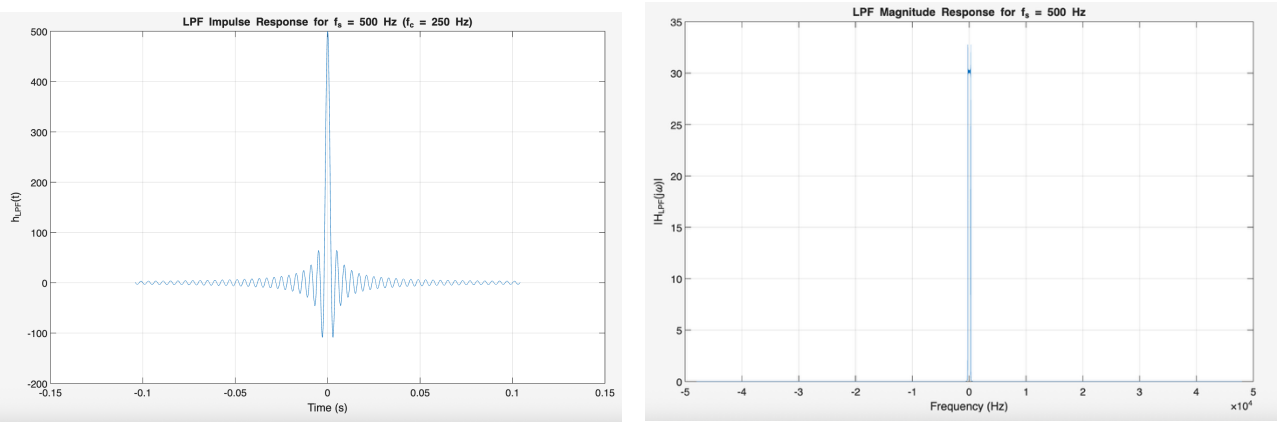

Figure 25: Plot for problem 1i

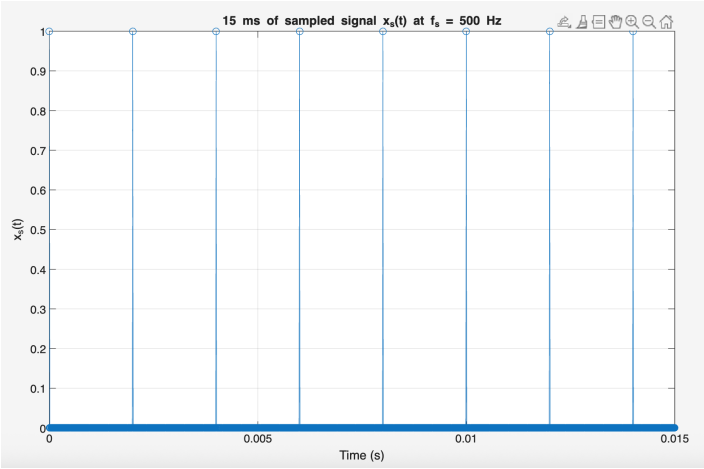

Figure 26: Plot for problem 1i

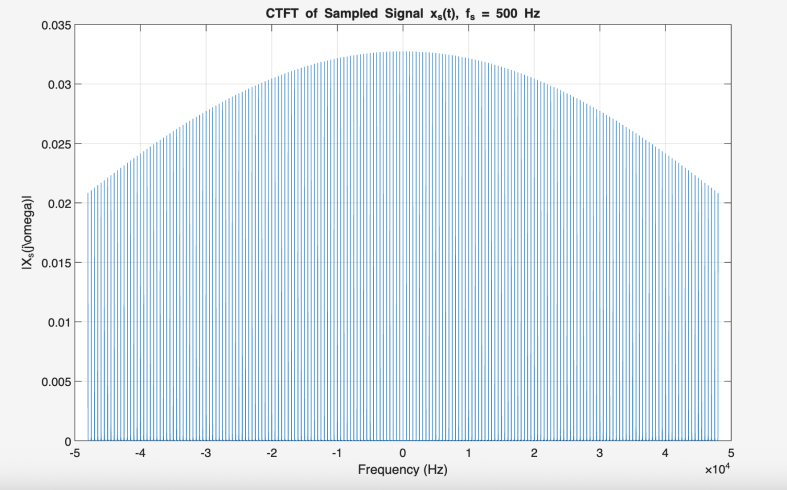

Figure 27: Plot for problem 1i

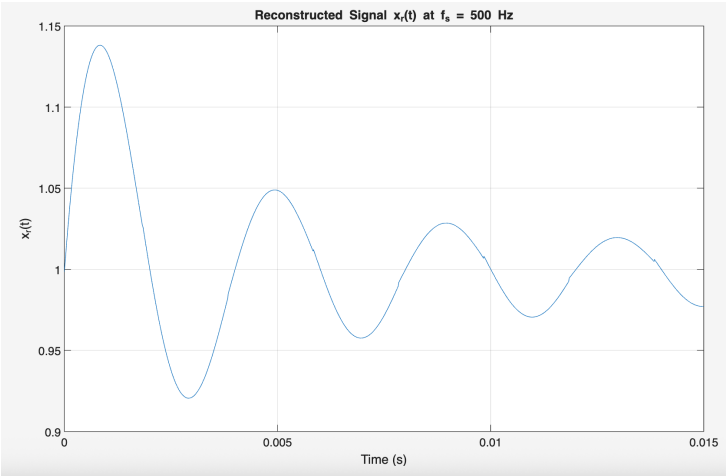

Figure 28: Plot for problem 1i

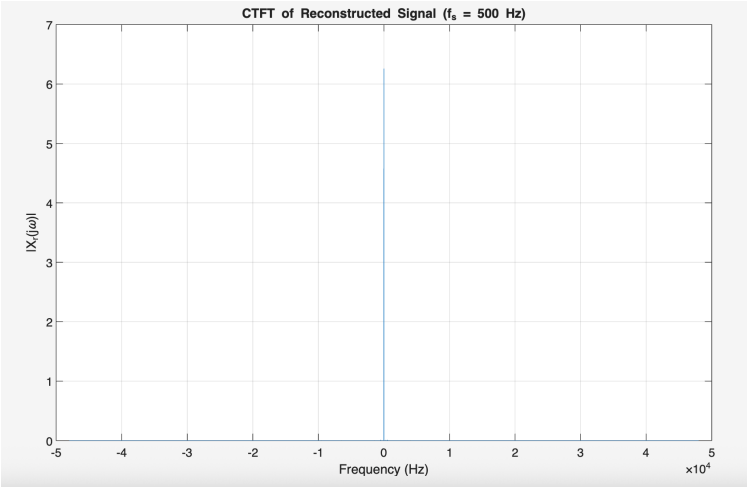

Figure 29: Plot for problem 1i

Using the same derivation as earlier if we substitute our value into our derived equation we get:

$$X_s(j\omega) = 500\pi \sum_{k=-\infty}^{\infty} (\delta(\omega - 2000\pi - 1000\pi k) + \delta(\omega + 2000\pi - 1000\pi k))$$

From the figures above we see that since the sampling frequency of 500Hz is below the Nyquist rate, which is 2000Hz, the recovered signal is very distorted. This is expected as the sampling rate is below of the Nyquist rate and the peaks of the x(t) signal as seen in figure 20 are not properly sampled as points are being skipped. Thus, when we want to recover it the signal, points will be missing. If we observe the derived equation, we notice that aliasing will occur as observed in figure 27.

To summarize, while the signal can be recovered at a sampling rate of 2000Hz, there is minor distortions with the amplitude, likely due to the LPF being very close to the boundary of the signal. However, at 500Hz the signal no longer resembles the original x(t).

Problem 2

(a)

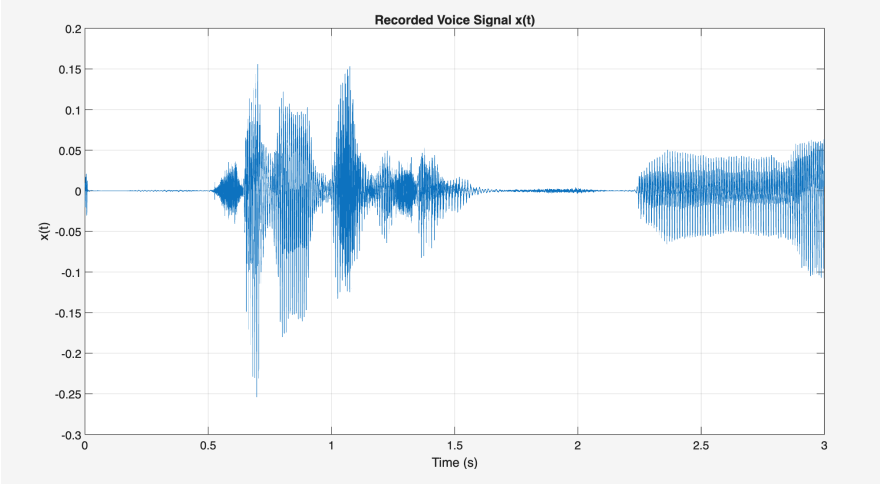

Figure 30: Plot for problem 2a

```
4   [x, Fs] = audioread('mp2_p2_original.wav');
5
6   % Part a)
7   t = (0:length(x)-1)/Fs;   % time vector
8
9   figure;
10  plot(t, x);
11  xlabel('Time (s)');
12  ylabel('x(t)');
13  title('Recorded Voice Signal x(t)');
14  grid on;
15
```

Figure 31: Code for problem 2a

The figures above show the plotted audio recording.

(b)

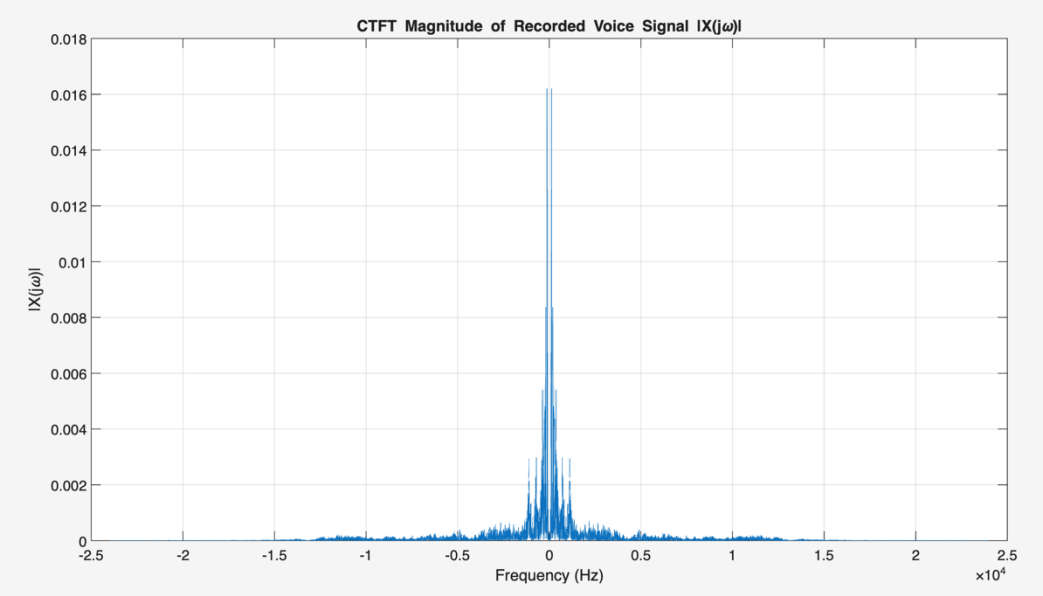

Figure 31: Plot for problem 2b

```
% Part b)
[X, w_axis, f_axis] = ctft(x', Fs);

figure;
plot(f_axis, abs(X));
xlabel('Frequency (Hz)');
ylabel('|X(j\omega)|');
title('CTFT Magnitude of Recorded Voice Signal |X(j\omega)|');
grid on;
```

Figure 32: Code for problem 2b

The figure above shows the magnitude of the Fourier transform. From figure 31 we can observe the bandwidth is approximately 4kHz since if we applied the -3dB rule from the peak magnitude, the range becomes too narrow. From the 4kHz point on both sides we see a slight noticeable gap thus this will be the selected bandwidth.

(c) The determined bandwidth was 4kHz. Therefore, the minimum Nyquist rate for my voice is 8kHz.

(d)

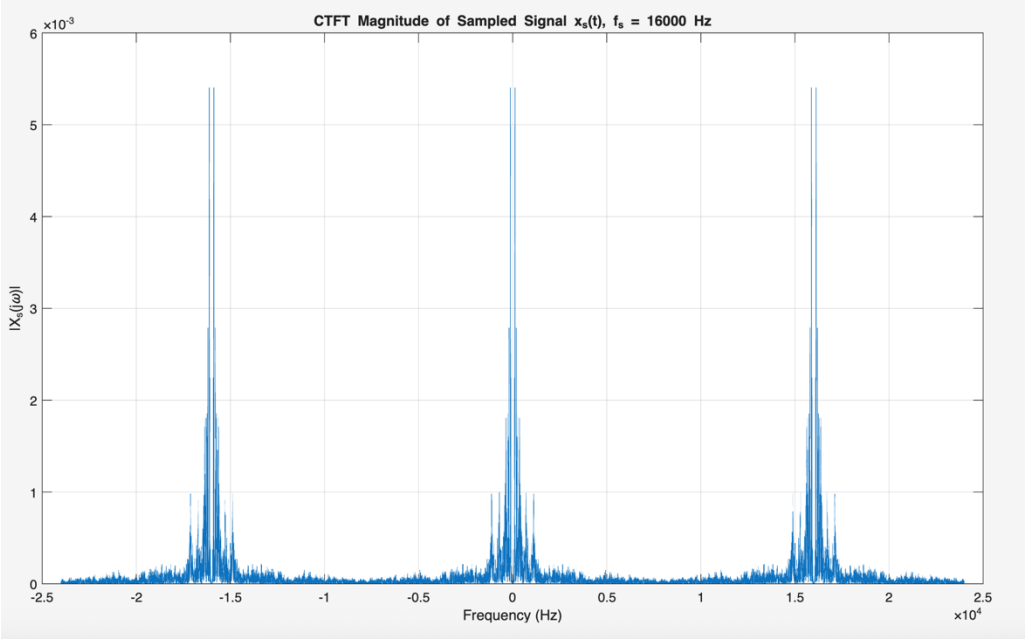

Figure 33: Plot for problem 2d

```
% Part d)
fs_samp = 16000;     % sampling rate
Ts = 1/fs_samp;

s = zeros(size(x'))';                    % Impulse train from Problem 1
sample_n = round((0:Ts:t(end)) * Fs) + 1;
sample_n(sample_n > length(x)) = [];
s(sample_n) = 1;

xs = x .* s;

[Xs, w_samp, f_samp] = ctft(xs', Fs);

figure;
plot(f_samp, abs(Xs));
xlabel('Frequency (Hz)');
ylabel('|X_s(j\omega)|');
title('CTFT Magnitude of Sampled Signal x_s(t), f_s = 16000 Hz');
grid on;
```

Figure 34: Code for problem 2d

The figure above shows the recorded audio sampled at a frequency of 16kHz. Using the impulse train from question 1, we see the X($j\omega$) spectrum repeated every 16kHz.

(e)

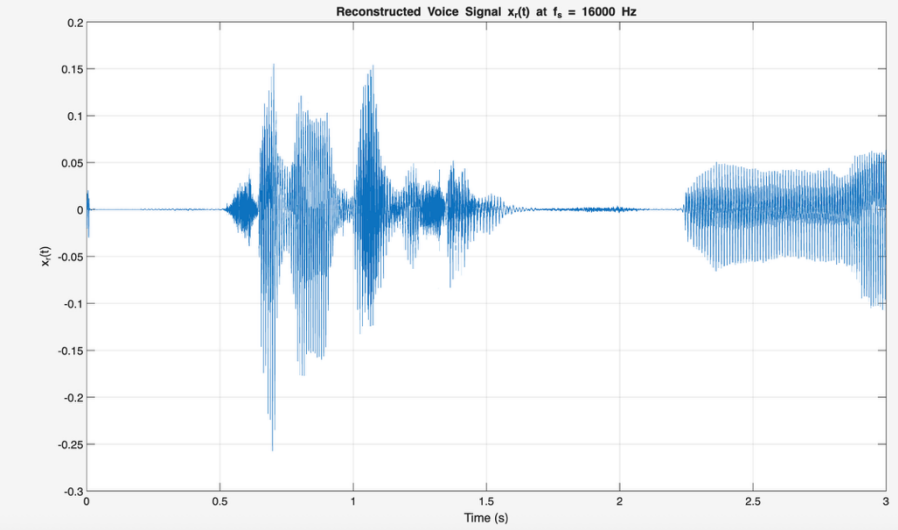

Figure 35: Plot for problem 2e

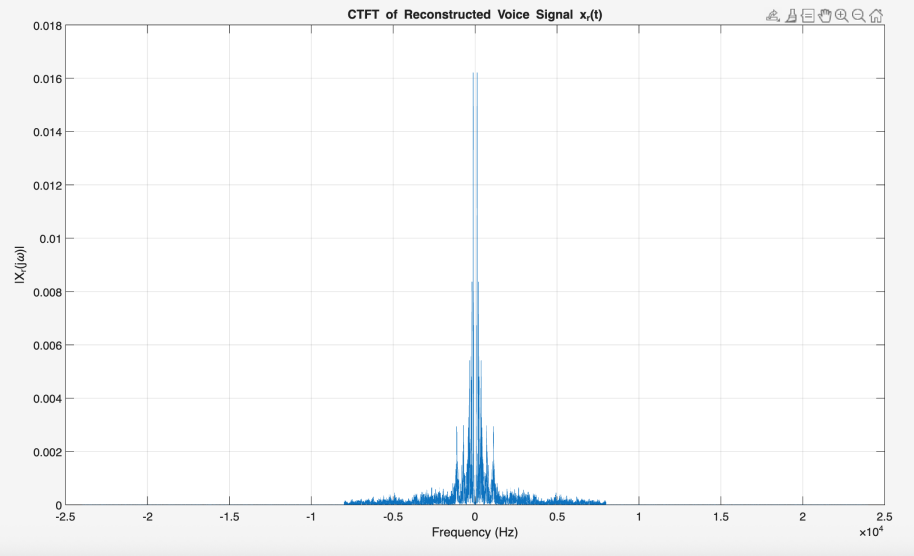

Figure 36: Plot for problem 2e

```
% Part e)
fc = 8000;          % This is half of the 16000 sampling freque
lpf_N = 20000;
lpft = linspace(-lpf_N/(2*Fs), lpf_N/(2*Fs), lpf_N);

hLPF = 2 * fc * sinc(2 * fc * lpft);    % this is the LPF from
[HLPF, w_lpf, f_lpf] = ctft(hLPF, Fs);

xr_full = conv(xs, hLPF);    % This section is identical to ho

mid = floor(length(hLPF)/2);
xr = xr_full(mid : mid + length(xs) - 1) * (1/fs_samp);

figure;
plot(t, xr(1:length(t)));
xlabel('Time (s)');
ylabel('x_r(t)');
title('Reconstructed Voice Signal x_r(t) at f_s = 16000 Hz');
grid on;

[Xr, w_r, f_r] = ctft(xr', Fs);

figure;
plot(f_r, abs(Xr));
xlabel('Frequency (Hz)');
ylabel('|X_r(j\omega)|');
title('CTFT of Reconstructed Voice Signal x_r(t)');
grid on;
```

Figure 36: Code for problem 2e

The figures above show the reconstructed signal which appears identical to the original signal with frequencies beyond 4kHz removed. When listening to the audio, it sounds the same however it sounds more bassy and deeper than the original audio file. Ignoring the deeper voice this is a faithful reconstruction. This also makes sense since the low pass filter is only allowing the lower frequencies to remain in the reconstructed signal resulting in the deeper sounding voice.

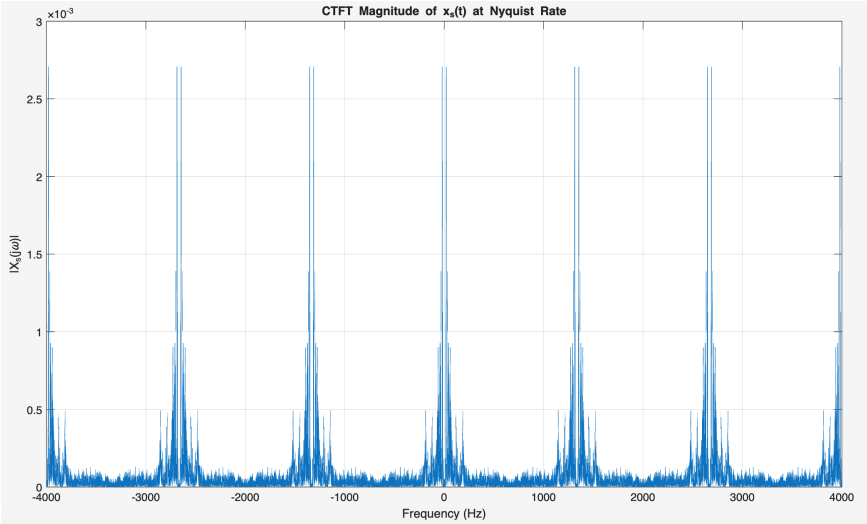

Figure 37: Plot for problem 2f

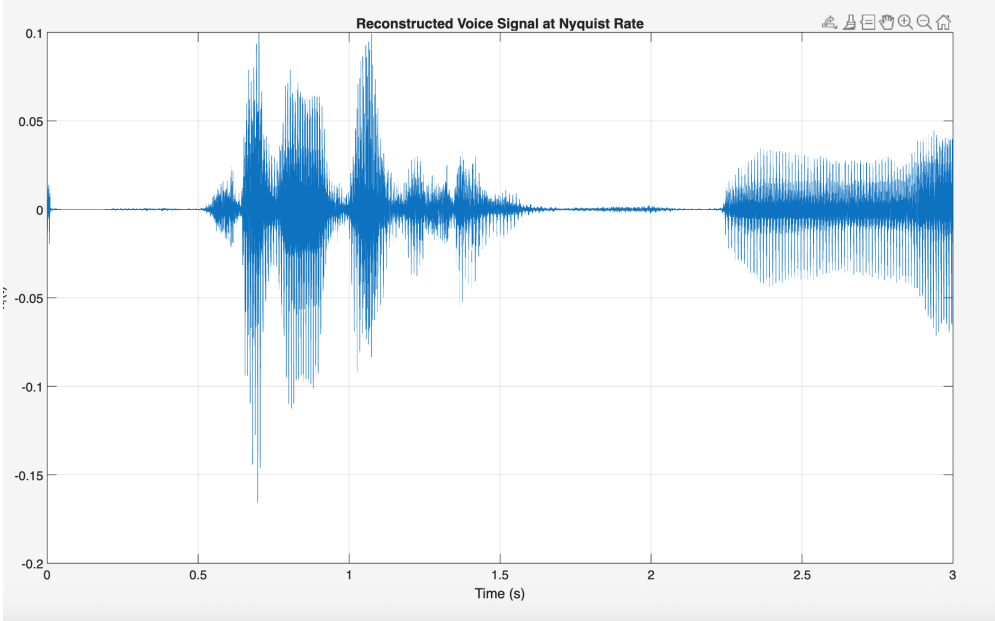

Figure 38: Plot for problem 2f

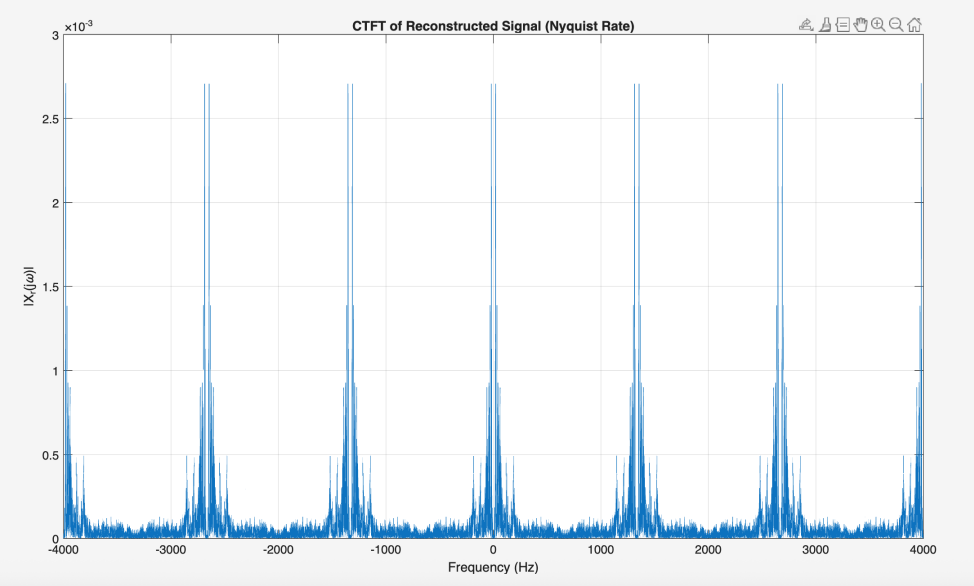

Figure 39: Plot for problem 2f

The figures above show the reconstructed signal which appears different than the original signal. The retrieved audio maintains a similar shape however it seems to have more oscillations (darker shade) in figure 38. This explains why the reconstructed audio sounds very distorted. This could be a result of picking an incorrect Nyquist rate, likely leading the audio signal to sound very harsh and broken. This is further justified by figure 39 which depicts aliasing.

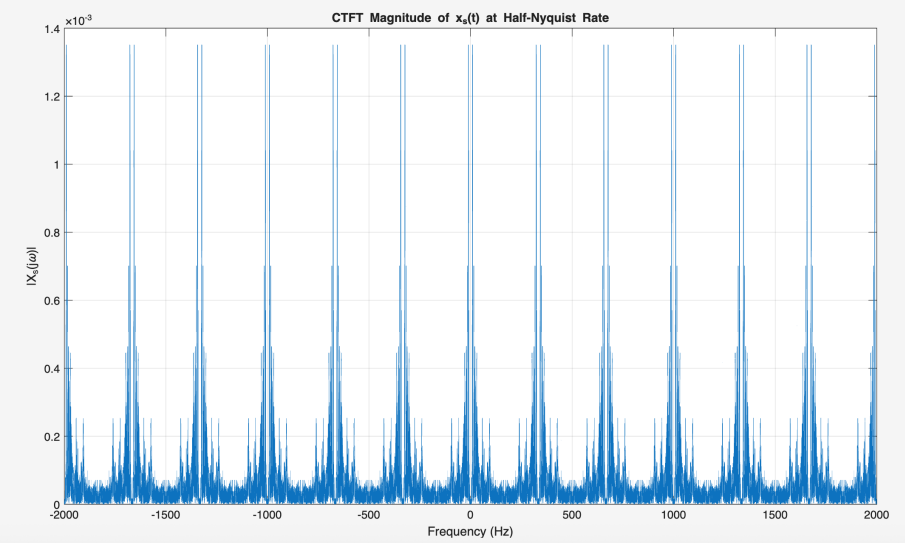

Figure 40: Plot for problem 2g

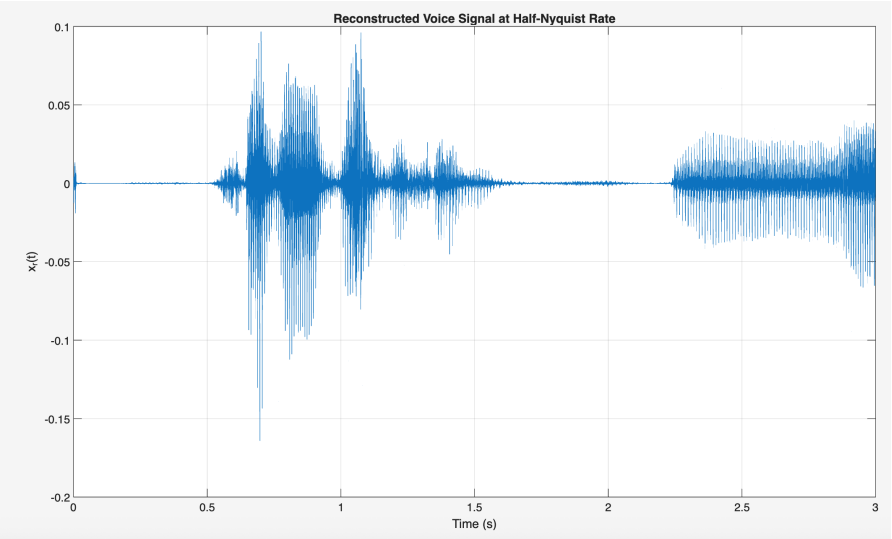

Figure 41: Plot for problem 2g

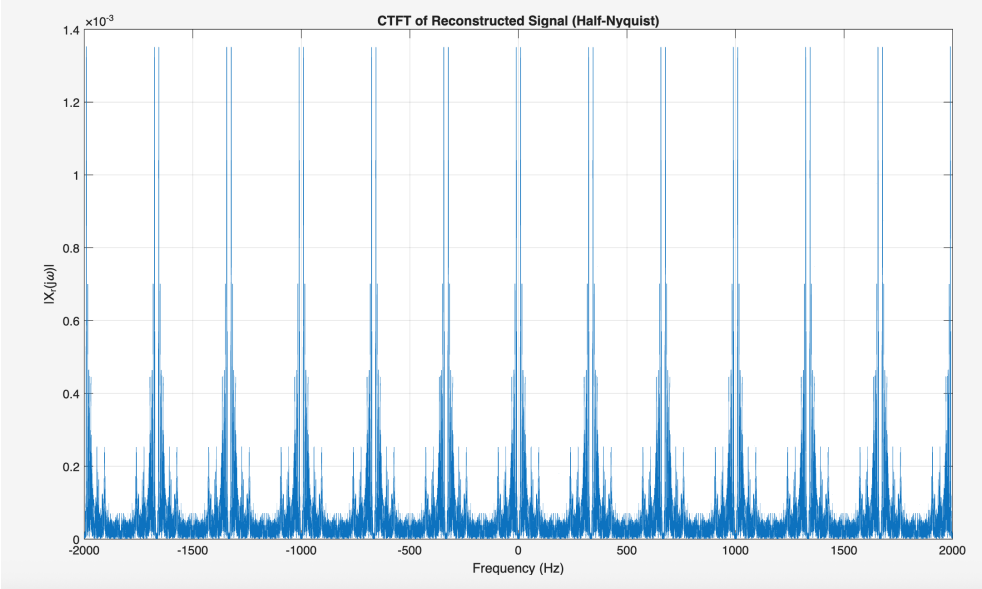

Figure 42: Plot for problem 2g

Similar to part f), this audio also sounds entirely broken and distorted due to aliasing visible in figure 42. This is expected as this is sampled at half the Nyquist rate.

Problem 3

(a)

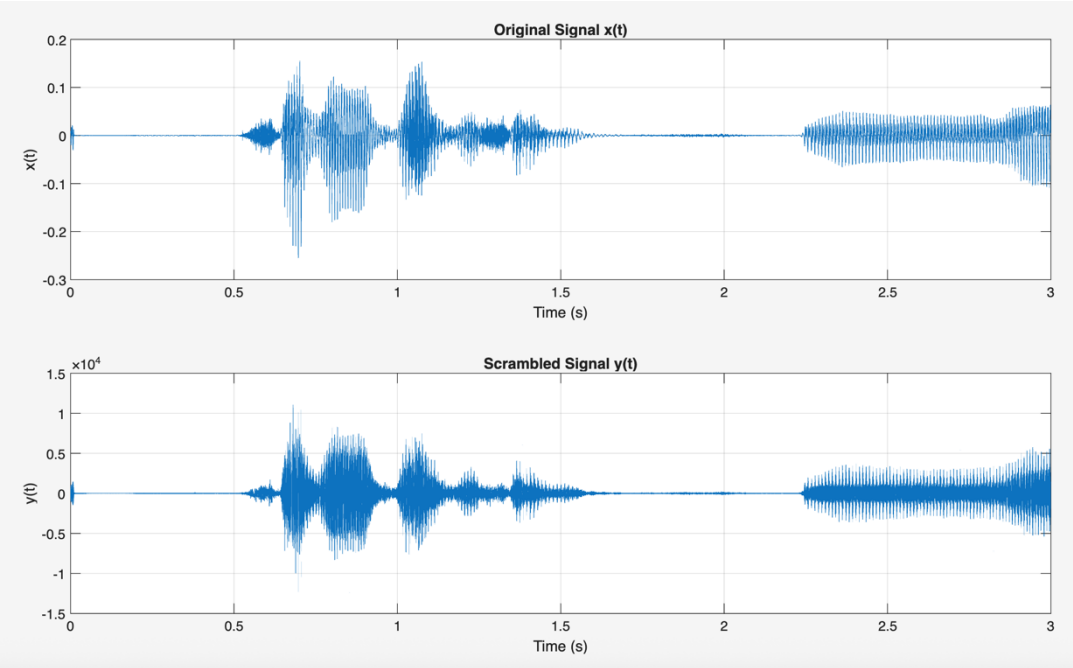

Figure 43: Plot for problem 3a

When comparing the 2 signals side to side, the scrambled signal looks identical in shape but seems to be of higher frequency due to it appearing darker indicating more oscillations. This agrees with how the audio sounds as it sounds like a high-pitched alien repeating the original audio. No words or humming can be recognized from the scrambled audio.

(b)

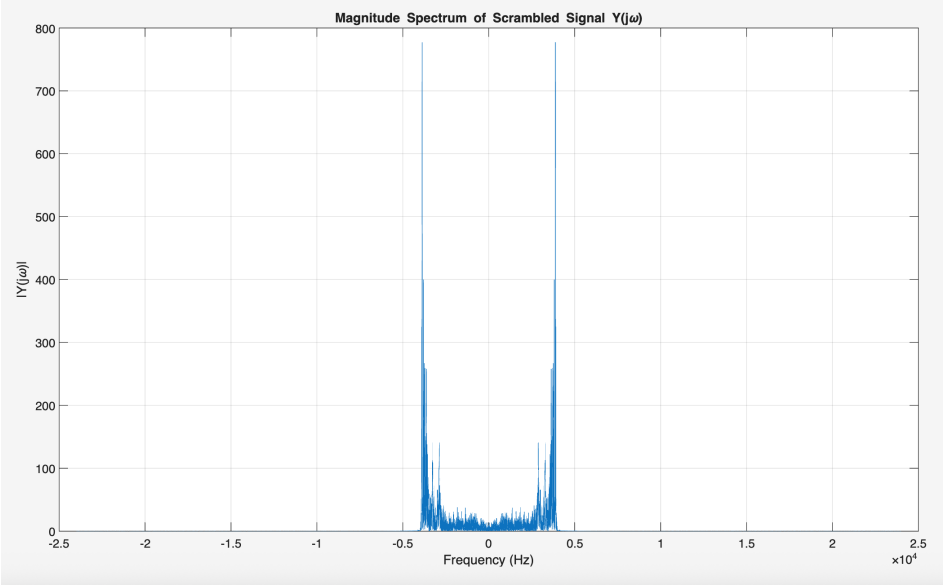

Figure 44: Plot for problem 3b

Figure 44 above shows the magnitude spectrum. It looks like the letter “U”. This is expected and makes sense, since the modulation shifts the spectrum to ±B, which in this case is at 4000Hz. Then the sinc() function acts like a LPF which removes anything beyond the 4000Hz point making it appear like the letter “U”.

(c)

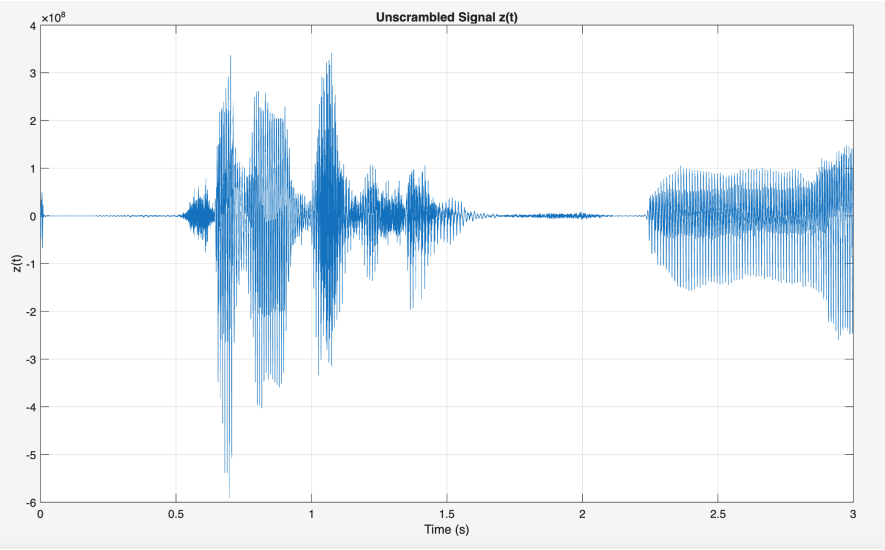

Figure 45: Plot for problem 3c

Figure 45 above shows the unscrambled signal, which looks very similar to the original signal. This also makes sense since it sounds very similar to the original audio, with a little bit of graininess to it. This could be due to some attenuation of frequencies when scrambling the signal since the sinc() removes frequencies beyond 4kHz.

(d)

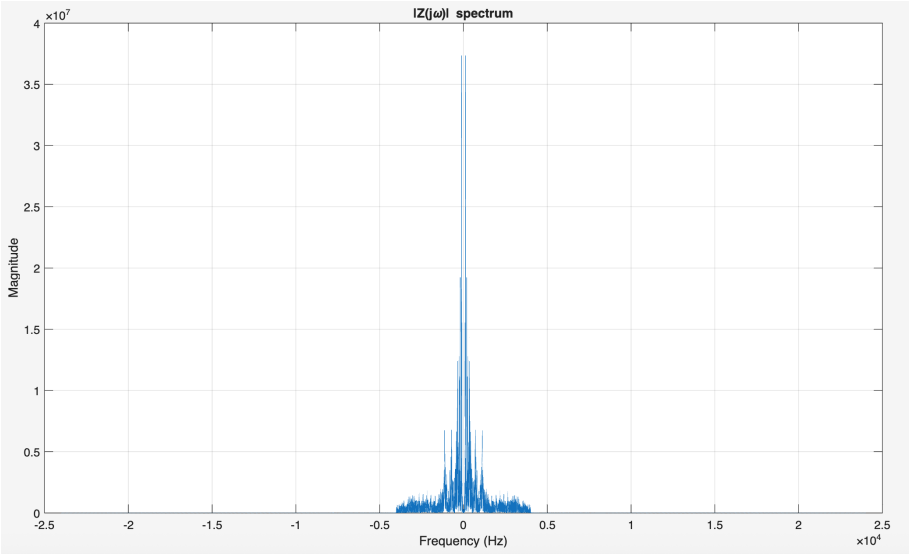

Figure 46: Plot for problem 3d

Figure 46 above looks identical to the spectrum of the original signal, except with all the frequencies at a higher magnitude. This explains why the audio sounds very similar but is not identical.