Acoustic Modumdum Project

Prisha Sadhwani and Manu Patil

11 May 2021

1 Introduction

An acoustic modem is a device that is used to transmit data underwater. It converts digital data into underwater sound signals. Those signals are then received by another modem, and are converted back into digital data. They have a multitude of practical applications, including underwater telemetry, remotely operated vehicle (ROV) and autonomous underwater vehicle (AUV) command and control, underwater monitoring and data logging, and other applications requiring underwater wireless communications.

Before any of that, however, the acoustic modem was designed for internet access. It modulated wave signals to encode digital information for transmission, and demodulated those signals to decode the transmitted information. The goal is to produce a signal that can be transmitted easily and decoded reliably to reproduce the original digital data, which is exactly what we're trying to do with this project.

We were given two .mat files, with different encoded messages in each. Our goal was to use what we learned throughout this course, such as convolution and lowpass filtering, to decode the messages and display the clean data.

2 Procedure

We decided to create a block diagram to accurately document the order of the steps we needed to follow, as shown in Figure 1.

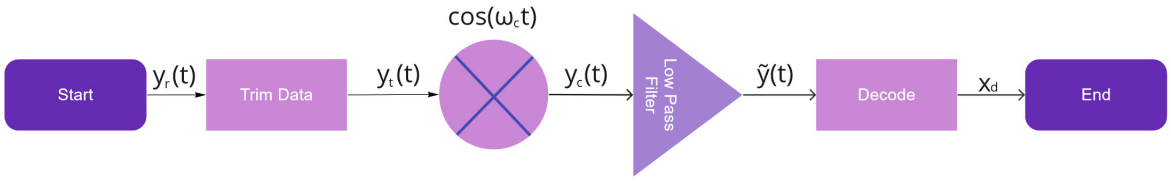
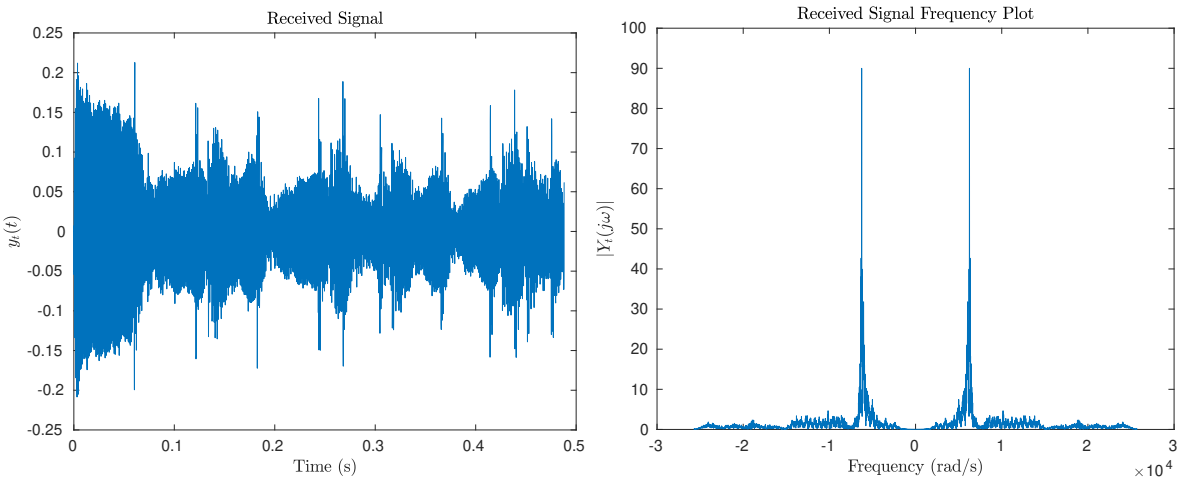


Figure 1: Block Diagram

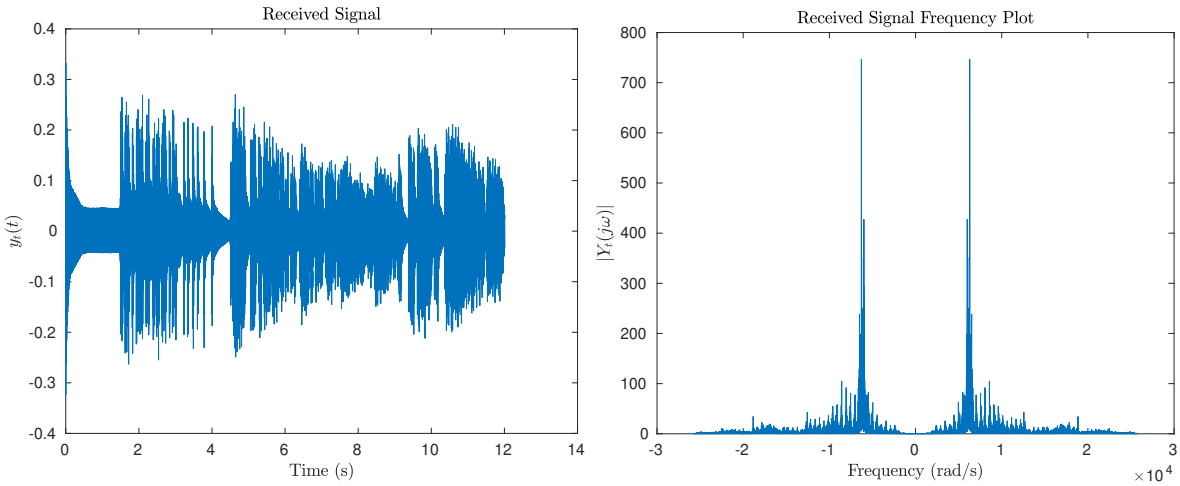
The first step we took was to load our signal. We were provided with two files, with each file including more samples than just the data. As a result, we selected the part of the signal that contained the data we were interested in, as shown in the graphs below.



(a) Short Signal Time Domain

(b) Short Signal Frequency Domain

Figure 2: Received Short Signal



(a) Long Signal Time Domain

(b) Long Signal Frequency Domain

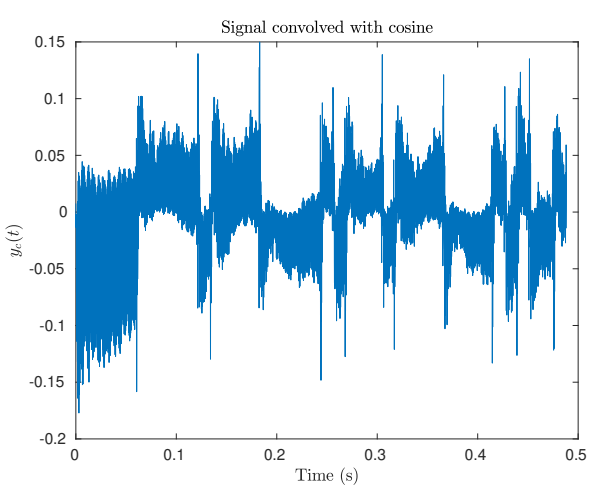
Figure 3: Received Long Signal

3 Convolution

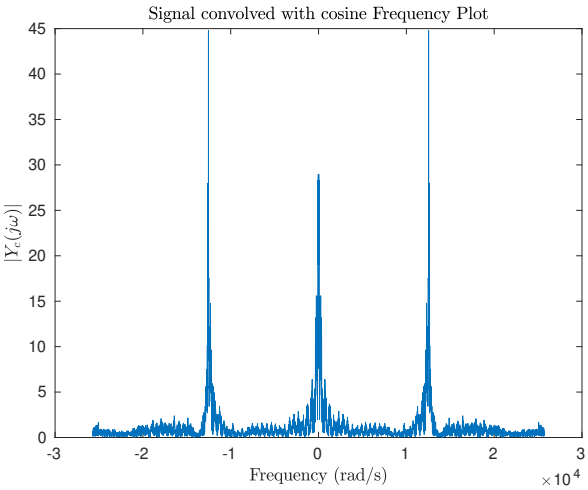
Next, we convolved our signal with Equation (1), which is the cosine function which serves as the carrier signal for our data, and plotted the results. This operation is a convolution in frequency which results in Equation (2), a multiplication in time.

$$c = \cos\left(2\pi \frac{f_c}{Fs} t\right) \tag{1}$$

$$y_c = y_t . * c \tag{2}$$

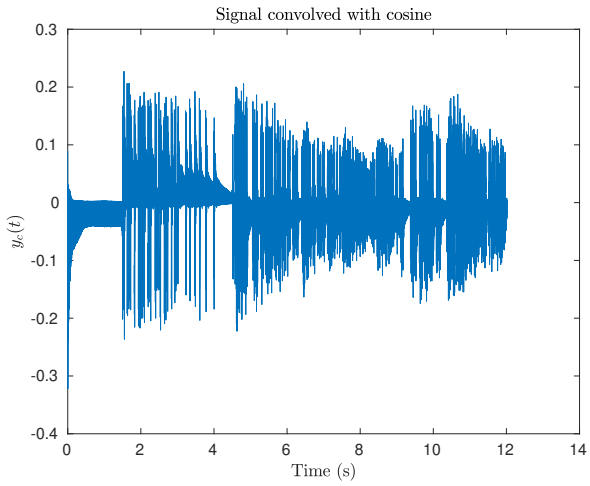


(a) Short Signal Time Domain

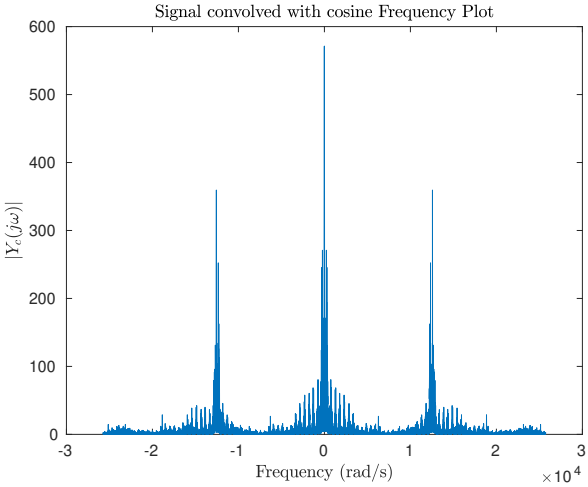


(b) Short Signal Frequency Domain

Figure 4: Convolved Short Signal



(a) Long Signal Time Domain



(b) Long Signal Frequency Domain

Figure 5: Convolved Long Signal

4 Filtering

From there, we set the cutoff frequency to $2\pi \cdot 1000 \frac{rads}{s}$, and created a 200 sample time vector to generate our sinc function, as shown in Equation (3), (4), (5). This operation is a convolution in time

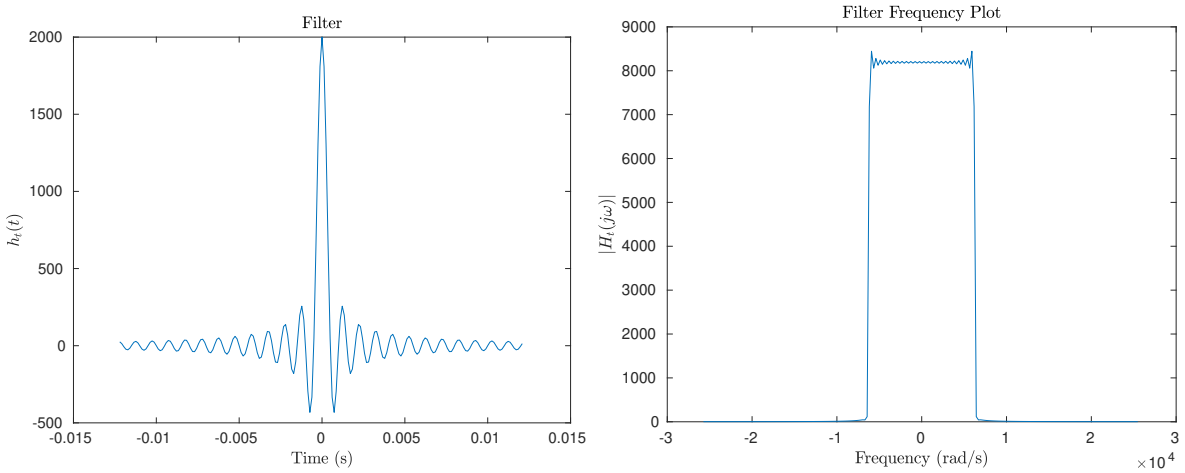
and as such we used the *conv* operator, with a graph of our filter being shown in Figure 6.

$$h = \frac{W}{\pi}\text{sinc}\left(\frac{W}{\pi}t\right) \tag{3}$$

$$W = 2\pi 1000 \tag{4}$$

$$t = (-100 : 1 : 99) * (1/Fs) \tag{5}$$

$$\text{conv}(y_c, h) \tag{6}$$

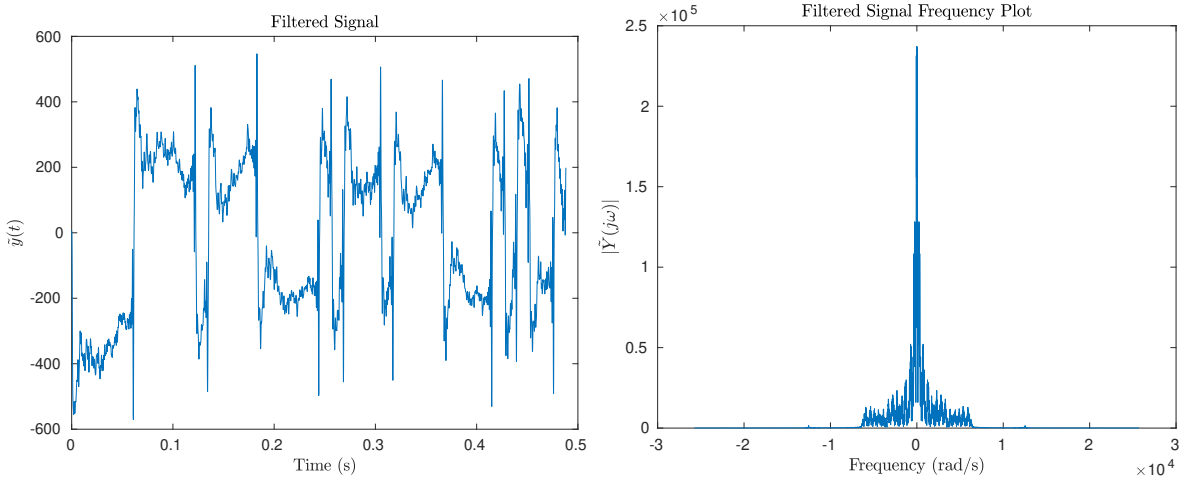


(a) Sinc Low Pass Filter Time Domain

(b) Sinc Low Pass Filter Freq Domain

Figure 6: Sinc Low Pass Filter

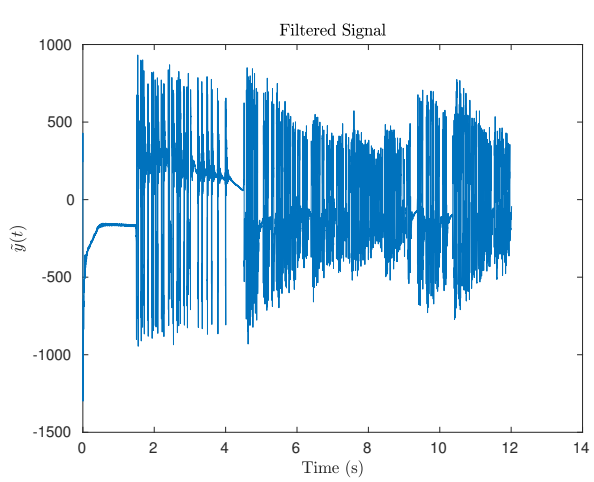
Figures 7 and 8 show our data with a sinc filter applied. Convolving our sinc function with our signal generated some extra data, so we had to truncate the filtered signal.



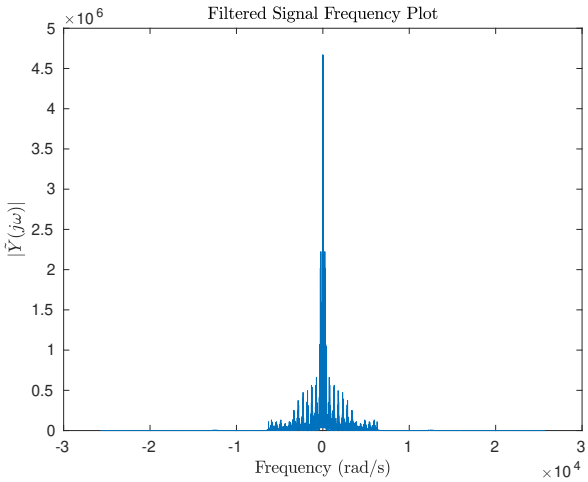
(a) Filtered Short Signal Time Domain

(b) Filtered Short Signal Freq Domain

Figure 7: Filtered Small Signal



(a) Filtered Long Signal Time Domain

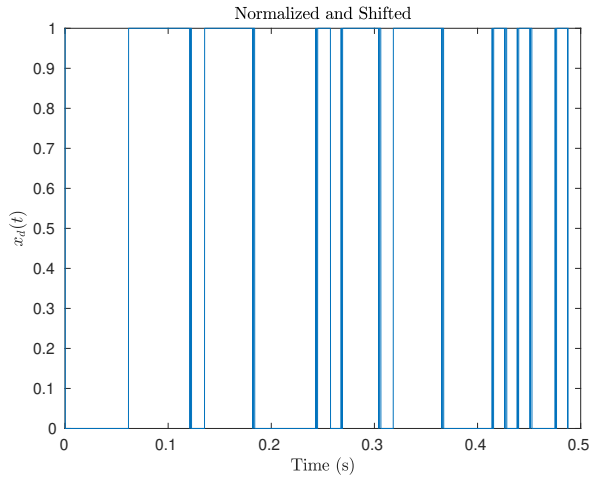


(b) Filtered Long Signal Freq Domain

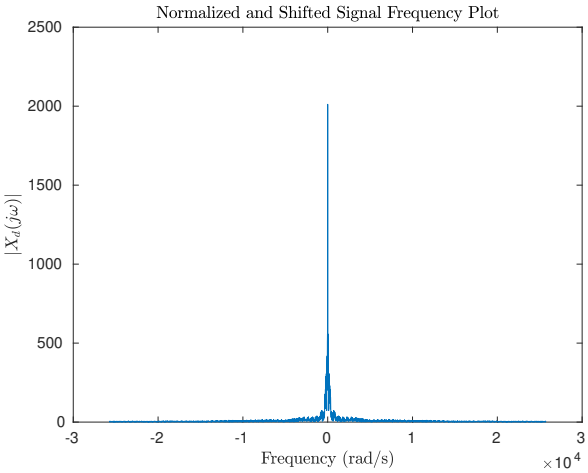
Figure 8: Filtered Long Signal

5 Results

From there, we cleaned up the data. We normalized the data to unit length, and shifted it from 1s and -1s to 1s and 0s. This allowed us to convert ASCII code into letters.

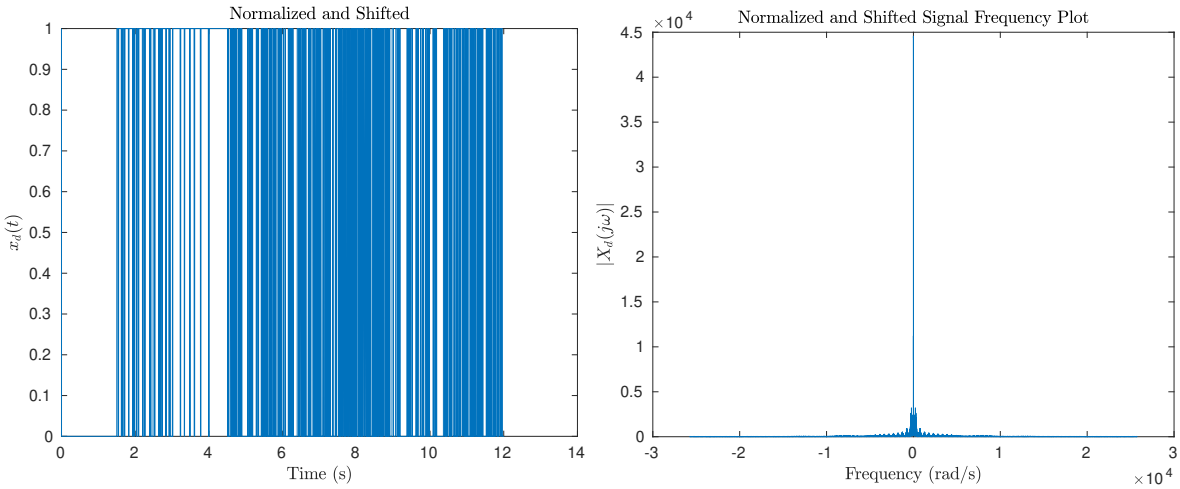


(a) Normalized and Shifted Short Time Domain



(b) Normalized and Shifted Short Frequency Domain

Figure 9: Normalized and Shifted Short Signal



(a) Normalized and Shifted Long Time Domain (b) Normalized and Shifted Long Frequency Domain

Figure 10: Normalized and Shifted Long Signal

Lastly, we converted our normalized data into a string that represented the decoded bits, and displayed the downsampled results. We found that the decoded short data read

> "Hello"

The decoded long data read

> "The answer to the Ultimate Question of Life, the Universe, and Everything is 42. The question to the ultimate answer is ..."

6 All Matlab code

The files are also attached separately for viewing convenience.

```
close all
clear

test = "long"; % Replace with "short" or "long" to test that file
should_plot = false; % "Turn plotting on or off

load(strcat(test,"_modem_rx.mat"))

% The received signal includes a bunch of samples from before the
% transmission started so we need discard the samples from before
% the transmission started.

start_idx = find_start_of_signal(y_r,x_sync);
% start_idx now contains the location in y_r where x_sync begins
% we need to offset by the length of x_sync to only include the signal
% we are interested in

y_t = y_r(start_idx+length(x_sync):end); % y_t is the signal which starts at the beginni
```

```
x_t = (0:1/8192:(length(y_t)-1)/8192)';

message_length = msg_length * 8 * 100;
x_t = x_t(1:message_length);
y_t = y_t(1:message_length);

if should_plot
    figure
    plot(x_t, y_t);
    title("Received Signal", 'Interpreter', 'Latex');
    xlabel("Time (s)", 'Interpreter', 'Latex');
    ylabel("$y_t(t)$", 'Interpreter', 'Latex');
    saveas(gcf,strcat('images/received_time_',test),'epsc')

    figure
    plot_ft_rad(y_t, Fs);
    title("Received Signal Frequency Plot", 'Interpreter', 'Latex');
    saveas(gcf,strcat('images/received_freq_',test),'epsc')
end
%% Convolve with cos
c = cos(2*pi*f_c/Fs * [0:message_length-1]');
y_c = y_t .* c;

if should_plot
    figure
    plot(x_t, y_c);
    title("Signal convolved with cosine", 'Interpreter', 'Latex');
    xlabel("Time (s)", 'Interpreter', 'Latex');
    ylabel("$y_c(t)$", 'Interpreter', 'Latex');
    saveas(gcf,strcat('images/convolved_time_',test),'epsc')

    figure
    plot_ft_rad(y_c, Fs);
    title("Signal convolved with cosine Frequency Plot", 'Interpreter', 'Latex');
    ylabel('$|Y_c(j\omega)|$', 'Interpreter', 'Latex');
    saveas(gcf,strcat('images/convolved_freq_',test),'epsc')
end
%% Low pass Filter

W = 2*pi*1000; % set the cutoff frequency to 2 pi * 1000 rads/s
t = (-100:1:99)*(1/Fs);  % create a 200 sample time vector to generate sinc
filter = W/pi*sinc(W/pi*t);
y_filtered = conv(y_c, filter);

% Truncate filtered signal by extra added by convolved cosine length
y_tilde = y_filtered(length(t)/2:end-length(t)/2);

if should_plot
    figure
    plot(t, filter);
    title("Filter", 'Interpreter', 'Latex');
```

```
    xlabel("Time (s)", 'Interpreter', 'Latex');
    ylabel("$h_t(t)$", 'Interpreter', 'Latex');
    saveas(gcf,'images/filter_time','epsc')

    figure
    plot_ft_rad(filter, Fs);
    title("Signal convolved with cosine Frequency Plot", 'Interpreter', 'Latex');
    ylabel('$|H_t(j\omega)|$', 'Interpreter', 'Latex');
    saveas(gcf,'images/filter_freq','epsc')

    figure
    plot(x_t, y_tilde);
    title("Filtered Signal", 'Interpreter', 'Latex');
    xlabel("Time (s)", 'Interpreter', 'Latex');
    ylabel("$\tilde{x}(t)$", 'Interpreter', 'Latex');
    saveas(gcf,strcat('images/filtered_time_',test),'epsc')

    figure
    plot_ft_rad(y_tilde, Fs);
    title("Signal convolved with cosine Frequency Plot", 'Interpreter', 'Latex');
    ylabel('$|\tilde{X}(j\omega)|$', 'Interpreter', 'Latex');
    saveas(gcf,strcat('images/filtered_freq_',test),'epsc')
end
%% Clean up
% Normalize data to unit length
y_norm = y_tilde ./ abs(y_tilde);
% and then shift to 0s and 1s
x_d = (y_norm + 1) ./2;

if should_plot
    figure
    plot(x_t, x_d);
    title("Normalized and Shifted", 'Interpreter', 'Latex');
    xlabel("Time (s)", 'Interpreter', 'Latex');
    ylabel("$\tilde{x}_n(t)$", 'Interpreter', 'Latex');
    saveas(gcf,strcat('images/normalshifted_time_',test),'epsc')

    figure
    plot_ft_rad(x_d, Fs);
    title("Normalized and Shifted Signal Frequency Plot", 'Interpreter', 'Latex');
    ylabel('$|\tilde{X}_n(j\omega)|$', 'Interpreter', 'Latex');
    saveas(gcf,strcat('images/normalshifted_freq_',test),'epsc')
end
%% Decode

% convert to a string assuming that x_d is a vector of 1s and 0s
% representing the decoded bits

% Uncomment to see all letters before downsampling
%BitsToString(x_d);
x_d = downsample(x_d, 100, 50);
```

```
data = BitsToString(x_d)
```