Creating a Receiver for an Acoustic Modem

Charlotte Ramiro de Huelbes and Lila Smith

April 2022

1 Introduction

The goal of this project was to implement a receiver for an acoustic modem. The system begins with a string message which is then converted into bits and up-sampled with a symbol period of 100, creating $m(t)$. This is then transmitted through an acoustic modem into $y_t(t)$, which we processed in our receiver to obtain the original message.

2 Method

A block diagram of our system is shown in Figure 1. The system begins with $m(t)$, which is then convolved with a high-frequency cosine wave and transmitted to give $y_t(t)$, the input to the receiver. The time and frequency domain graphs of y_t are shown in Figure 2.

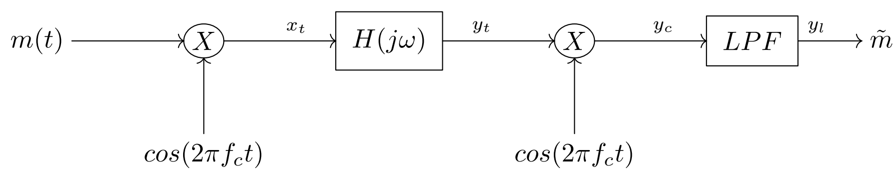


Figure 1: Block diagram of the entire system. Note that we only aimed to implement the receiver, thus our project is implemented starting with y_t.

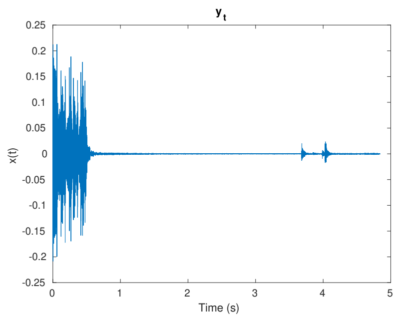
Our receiver takes this signal and convolves it in the frequency domain with the same high frequency cosine wave. The resulting signal y_c is shown in Figure 3, while the equations for this are seen below in both the frequency and time domains.

$$Y_c(j\omega) = Y_r(j\omega) * \pi[\delta(\omega - \omega_0) + \delta(\omega + \omega_0)]$$
$$y_c(t) = y_r(t) \times cos(2\pi f_c t)$$

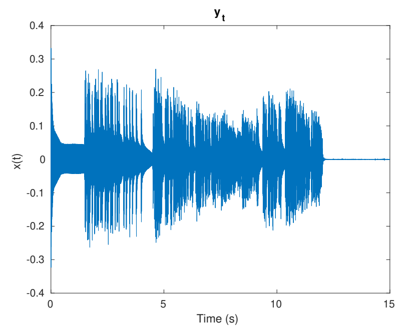
It then multiplies it in the frequency domain through a low pass filter with a cutoff of $\frac{f_c}{2}$ in order to retrieve the original signal $\tilde{m}(t)$. Since this is convolution in the time domain, we convolve $y_c(t)$ with a sinc function (ideal lowpass filter in the time domain). This resulting signal is shown in Figure 4, while the equation in the time domain is below.

$$\tilde{m}(t) = y_r(t) * \frac{f_c}{2\pi}\,\text{sinc}(\frac{f_c}{2\pi t})$$

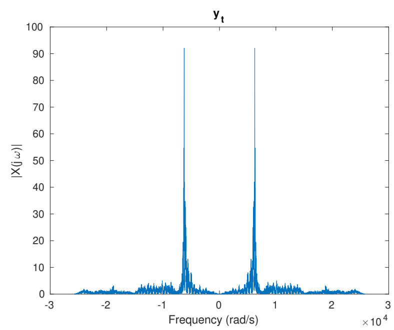
We then used the originally implemented symbol period in order to decode the bits from the signal. We did this by averaging 100 elements of the $\tilde{m}(t)$ matrix at a time (as the symbol period is 100) and converting them into bits based on if the average value was above or below zero (1 for above and 0 for below). A representation of this can be seen in Figure 5.



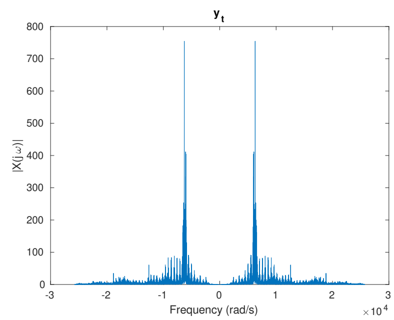
(a) Time domain graph of short message



(b) Time domain graph of long message

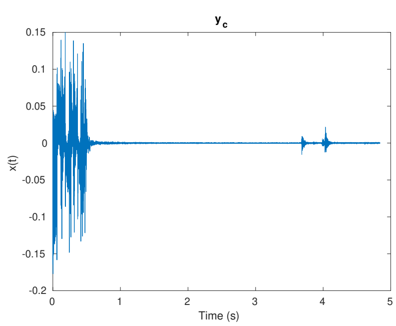


(c) Frequency domain graph of short message

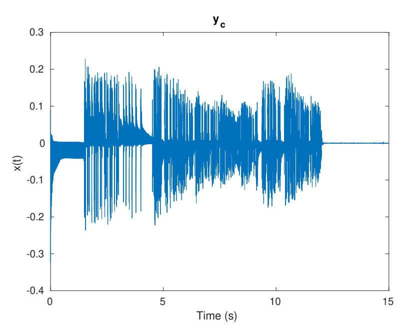


(d) Frequency domain graph of long message

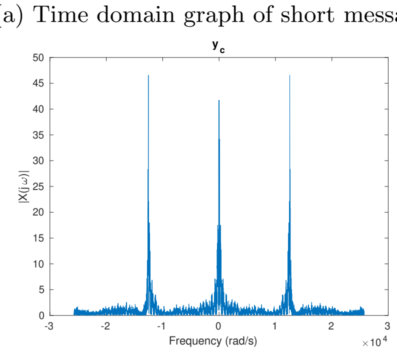
Figure 2: y_t: The signal transmitted by the modem, synced to beginning.



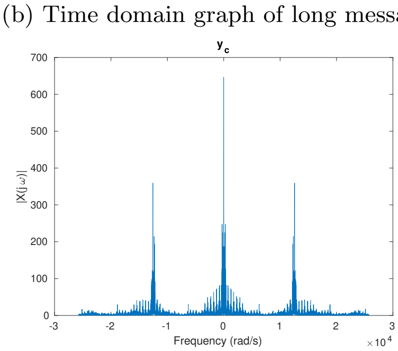
(a) Time domain graph of short message



(b) Time domain graph of long message

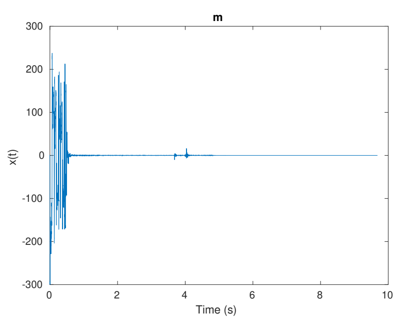


(c) Frequency domain graph of short message

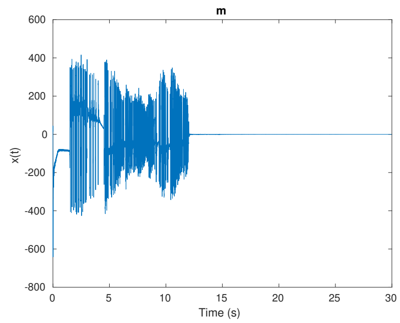


(d) Frequency domain graph of long message

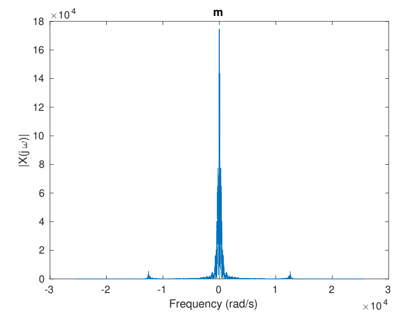
Figure 3: y_c: While the signal looks similar to y_t in the time domain, after convolving with $\cos(2\pi f_c t)$, we see that in the frequency domain there are now three peaks instead of two.



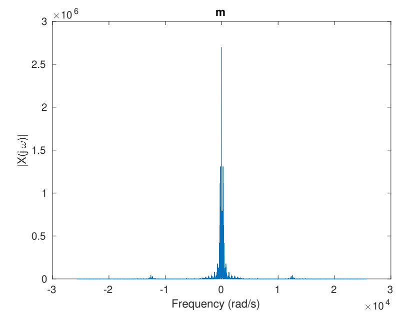
(a) Time domain graph of short message



(b) Time domain graph of long message



(c) Frequency domain graph of short message

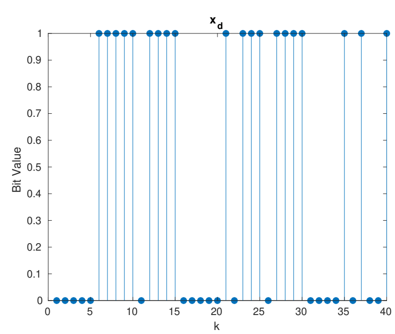


(d) Frequency domain graph of long message

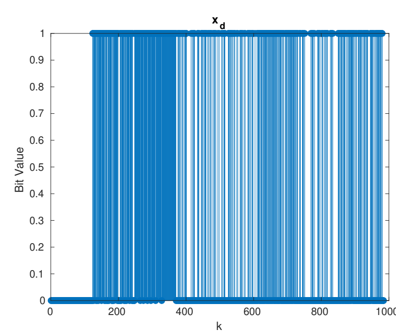
Figure 4: m: After applying the lowpass filter with cutoff of $\frac{f_c}{2}$, the signal is much more block-like in the time domain. We see that in the frequency domain the three peaks have been reduced down to one.

3 Results

Our receiver was able to accurately decode both the short and long messages provided. You can find a video of this here [1].



(a) Time domain graph of short message



(b) Time domain graph of long message

Figure 5: Binary representations of signal after averaging symbol periods to see if they are above zero (so bit is 1) or below zero (so bit is 0).

[1]Full text link: https://youtu.be/hvkZI1XzYns

A Code Listing

```
load long_modem_rx.mat

% The received signal includes a bunch of samples from before the
% transmission started so we need discard these samples that occurred
% 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 beginning of the transmission

% Multiply with the same cosine function to recenter original function
% We multiply because we want to convolve in the frequency domain
t = 0:(1/Fs):(length(y_t)-1)/Fs;
c = cos(2*pi*f_c*t);
y_c = c .* y_t';

% Use a lowpass filter to filter high frequencies created with cosine
% We convolve because we want to multiply in frequency domain
W = 0.5*f_c;
h_lowpass = (W/pi)*sinc(W/pi*t);
y_l = conv(y_c, h_lowpass);

% Find average highs and lows per symbol period for message length
x_d = zeros([msg_length*8, 1]);
for i=1:length(x_d)
    average = mean(y_l((i-1)*100+1:i*100));
    x_d(i) = average > 0; % Write a 1 if co
end


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