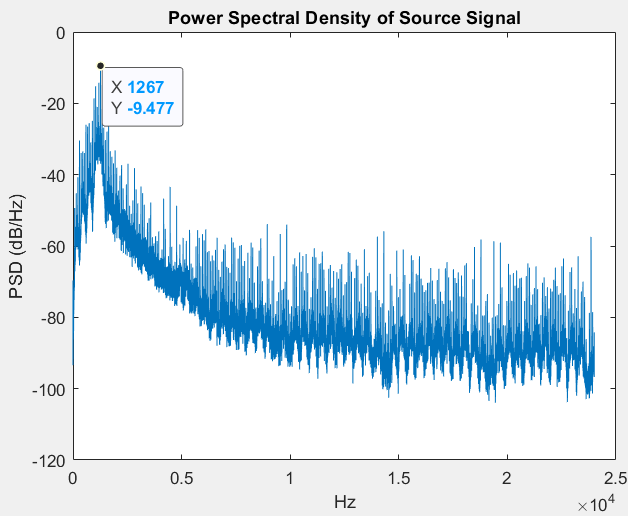
# Step 01

## Get the spectrum of the transmitted QPSK waveform, and the error performance without the error correction coding in place.

*Power Spectral Density of QPSK Waveform:*



Without error correcting code, I was able to receive a transmission over wire with only 2 bit errors, and a transmission over the air with 17 bit errors.

# Step 02

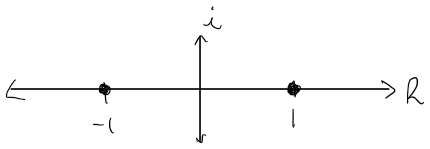
## Design your new MSK modulation.

We will attempt to continue to use a baud of 300. Our constellation will be binary, so the number of symbols in our symbol set is 2, and our baud is equal to our bit rate of 300 bits/second.

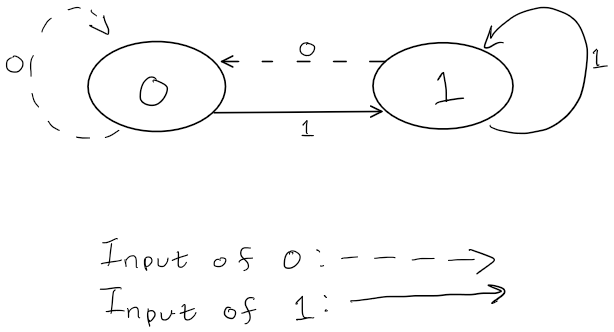
## What data rate do you expect to use with this bandwidth? What is the limit on this rate?

The limit is a data rate of 300 bits/second.

## What is the constellation of this modulation?



## What is the state diagram of this modulation?



## How do you intend to construct the modulation? (There are two approaches that appear quite different but produce identical waveforms!)

I intend to construct this modulation from frequency shift modulation. I will calculate the phase constraint for each symbol time.

## Implement this.

I was able to successfully implement my modulation technique. The time domain plot is shown below. This shows the symbol periods for a bitstream of [0 1 0 0] (first four bits of ‘A’ in hexadecimal). I have also pasted the code for the function that encompasses the MSK modulation. This function is called modulateMSK().

*Matlab Code for modulateMSK():*

function [s, t] = modulateMSK(bitstream, fc, baud, Fs)

%modulateMSK Summary of this function goes here:

% This function takes a stream of bits 'bitstream', and modulates a carrier

% waveform with a frequency 'fc' using Minimum Shift Keying, and sampled

% with a sampling frequency of 'Fs'.

% Inputs:

% bitstream: vector containing the bits that will be modulated to an

% MSK waveform

% fc: carrier frequency of the waveform

% baud: baud of modulation (symbols/second)

% Fs: sampling frequency

% Outputs:

% s: vector containing the values of the modulated source signal

% t: time vector of the source signal s

Dk = 2\*bitstream - 1; % Convert bitstream to -1s and 1s

Dk = [-1 Dk]; % we need an initial value for Dk

N = length(bitstream);

T = 1/baud;

t = 0:1/Fs:N\*T-(1/Fs);

% == Calculate Phase Constraint Vector, Xk ==

Xk = zeros(1, N+1);

for k = 2:N+1

Xk(k) = mod(Xk(k-1) + ((pi\*(k))/2)\*(Dk(k-1) - Dk(k)), 2\*pi);

end

Xk = Xk(2:length(Xk));

Dk = Dk(2:length(Dk));

% == Generate MSK Waveform ==

symbolIndex = floor(t/T) + 1;

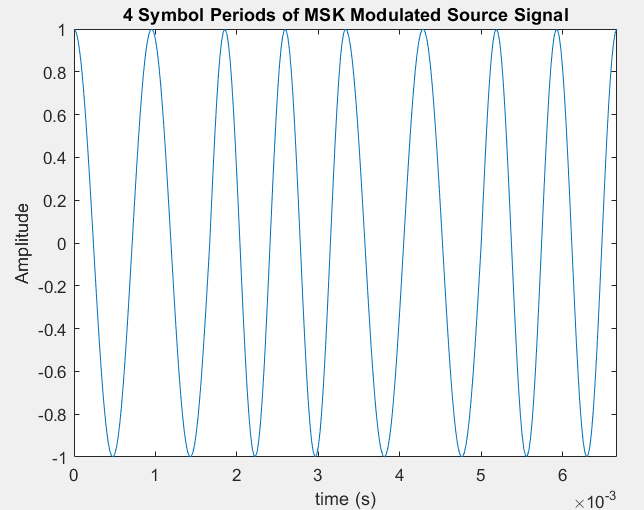
Xk = Xk(symbolIndex);

Dk = Dk(symbolIndex);

s = cos(2\*pi\*(fc + Dk/(4\*T)).\*t + Xk);

end

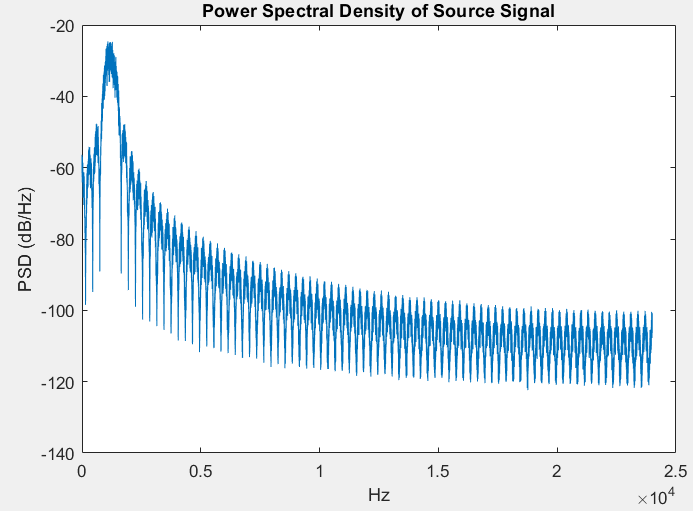
*Time Domain Plot of Modulated MSK Waveform:*



## Capture the spectrum of the transmitted signal, both across the band and close in to several hundred Hz each side of the carrier frequency. How does this spectrum compare with your QPSK waveform spectrum from Lab #3?

I converted the entire Isaiah passage to bits then modulated it to an MSK waveform. After that, I captured the spectrum. The spectrum of the entire source signal is shown below.

*Power Spectral Density Plot of Modulated Source Signal:*



From this plot, and the plot shown earlier of the PSD of the QPSK modulated waveform, we can see that the MSK waveform has a slightly wider main lobe, however, it decreases quickly. The PSD of the higher and lower frequencies outside the bandwidth is significantly smaller for the MSK waveform than for the QPSK waveform.

# Step 03

## Decide on a decoding approach and implement it.

We will use a matched filter approach to decoding. We can correlate expected waveforms to the received waveform to implement this.

*Matlab Code for demodMSK():*

function [bitstream] = demodMSK(s, fc, baud, Fs)

%demodMSK Summary of this function goes here:

% This function demodulates a waveform that was modulated using Minimum

% Shift Keying into a stream of bits.

% Inputs:

% s: vector containing the values of the modulated source signal

% fc: carrier frequency of the waveform

% baud: baud of modulation (symbols/second)

% Fs: sampling frequency

% Outputs:

% bitstream: vector containing the demodulated bits

T = 1/baud;

samplesPerSymbol = T\*Fs;

N = length(s);

s1 = modulateMSK([1], fc, baud, Fs); % Waveform of an MSK modulated 1 bit

s0 = modulateMSK([0], fc, baud, Fs); % Waveform of an MSK modulated 0 bit

bitstream = logical.empty;

for n = 1:samplesPerSymbol:N

bit = 0;

corr1 = xcorr(s(n:n+samplesPerSymbol-1),s1); % correlate s1

max1 = max(corr1);

min1 = min(corr1);

corr0 = xcorr(s(n:n+samplesPerSymbol-1),s0); % correlate s0

max0 = max(corr0);

min0 = min(corr0);

if((max1 + abs(min1)) > (max0 + abs(min0)))

bit = 1;

end

bitstream = cat(2, bitstream, bit);

end

end

## Explain the decoding approach you have chosen to use. Is it an optimal approach?

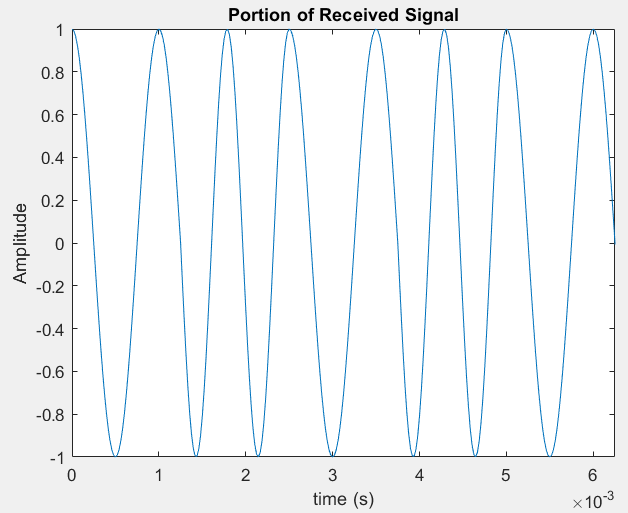
The decoding algorithm uses two matched filters to determine what bit was modulated for the symbol period. Since there can be technically 4 different waveforms for MSK modulation during a symbol period, we will be using two matched filters, but looking at both the minimum and maximum correlation of the outputs of the filter. The upper and lower frequencies can be shifted in their phase by π or 0, so a matched filter output of the same frequency would produce a near perfect correlation, or a nearly completely uncorrelated output at the minimum or maximum peaks. We can determine which waveform was sent using only two filters by looking at the maximum and minimum peaks.

This is an optimal approach, as it is a simple algorithm, with a theory that works well. (I have tested the demodulation algorithm as of writing this.)

## Why did you select the decoding approach you did?

I selected this decoding approach, as it was easy to implement, and is not too difficult to grasp conceptually.

## Working on a file



*Matlab Console Output:*

Who has believed our message and to whom has the arm of the Lord been revealed? 2 He grew up before him like a tender shoot, and like a root out of dry ground. He had no beauty or majesty to attract us to him, nothing in his appearance that we should desire him.3 He was despised and rejected by mankind, a man of suffering, and familiar with pain. Like one from whom people hide their faces he was despised, and we held him in low esteem. 4 Surely he took up our pain and bore our suffering, yet we considered him punished by God, stricken by him, and afflicted. 5 But he was pierced for our transgressions, he was crushed for our iniquities; the punishment that brought us peace was on him. and by his wounds we are healed. 6 We all, like sheep, have gone astray, each of us has turned to our own way; and the Lord has laid on him the iniquity of us all.

receivedText =

'Who has believed our message and to whom has the arm of the Lord been revealed? 2 He grew up before him like a tender shoot, and like a root out of dry ground. He had no beauty or majesty to attract us to him, nothing in his appearance that we should desire him.3 He was despised and rejected by mankind, a man of suffering, and familiar with pain. Like one from whom people hide their faces he was despised, and we held him in low esteem. 4 Surely he took up our pain and bore our suffering, yet we considered him punished by God, stricken by him, and afflicted. 5 But he was pierced for our transgressions, he was crushed for our iniquities; the punishment that brought us peace was on him. and by his wounds we are healed. 6 We all, like sheep, have gone astray, each of us has turned to our own way; and the Lord has laid on him the iniquity of us all.'

>>

## Over Wire

I had some issues with receiving the data over the wire initially. I think this is because I am not syncing the characters properly, and decoding over the wrong set of bits to retrieve the characters. I will try to write some code to sync the character bits and then decode the bits to text.

After some testing, I have come up with a way to properly align the bits and get them decoded to characters. I correlated the header bits with the demodulated bitstream that we received, and from here, I can get the bit offset of where the header starts. I can start decoding to text from this bit. It worked fairly well, and I will show the code that syncs the bits. It is in a function called syncBits(). A description of the function is provided via comments in the function’s code. I used a shorter string of just my name ‘Christopher David Hansen’ to test my new code and method.

*Matlab Code for syncBits():*

function [sampleOffset] = syncBits(headerBits, rxBitstream)

% This function takes a bitstream and bits of a header that has been

% added to the front of the bitstream and syncs the header and bitstream

% together so that when the bitstream is decoded to text, the characters

% are synchronized. The function provides the offset for the decoding to

% start as the output.

% Inputs:

% headerBits: preamble bits of the bitstream

% rxBitstream: bitstream to be synced with header and then

% to be converted to text.

% Outputs:

% sampleOffset: bit offset to start decoding to text

[headerCorr, lags] = xcorr(rxBitstream,headerBits);

figure();

plot(lags, headerCorr);

title('Correlation of Header Bits and Rx Bitstream');

[~, offset] = max(abs(headerCorr));

sampleOffset = lags(offset+1)

end

*Matlab Console Output (Over Wire):*

Header Sample Offset

headerSampleOffset =

66194

sampleOffset =

1

receivedText =

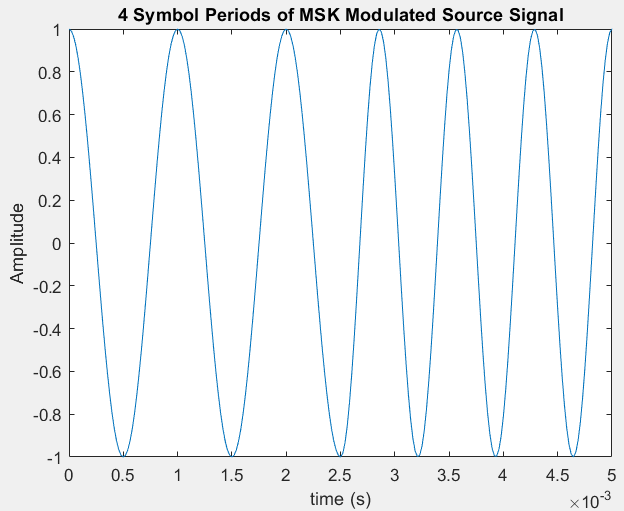
' ÿÿ Christopher David Hansen ’

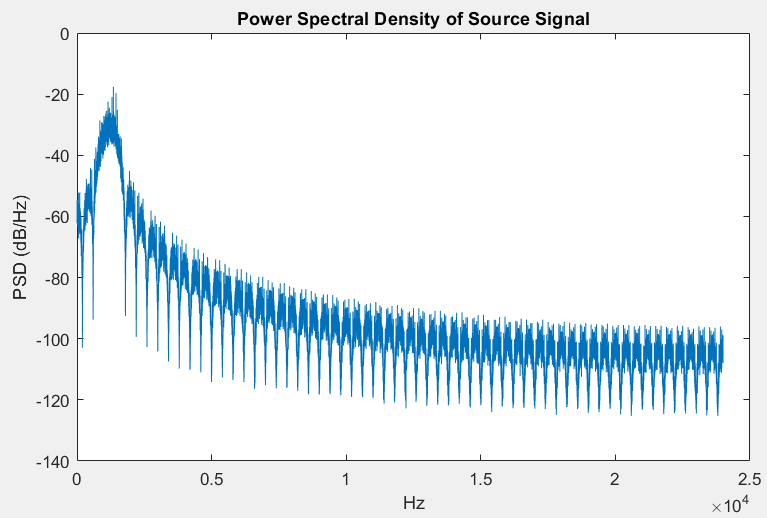
As we can see from the received text, I was able to receive my entire name without error. I will move on to the entire Isaiah passage.

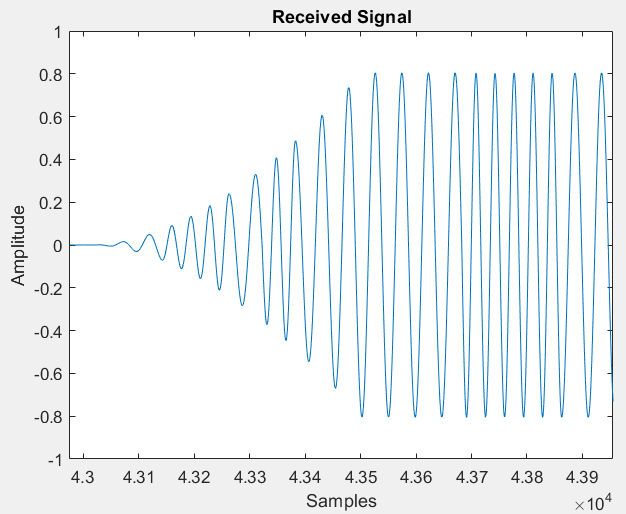
After trying to receive the entire Isaiah passage, I had issues with the output having errors at the very end of the passage. The last 8 characters from the passage had bit errors, so I am going to try to implement my encoding and decoding algorithms from last lab to try and fix this issue.

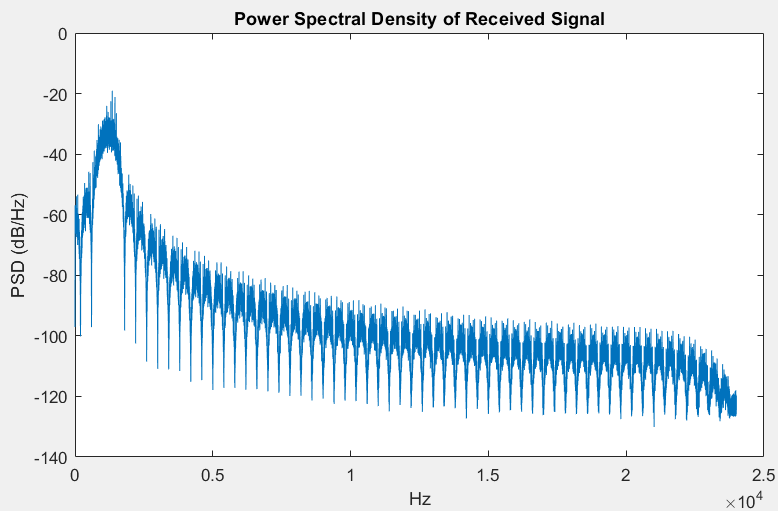
There was some fine tuning I had to make to my code, but I was able to incorporate my encoding and decoding functions from last lab to get a completely error free transmission of the entire Isaiah passage over wire. I sent and received over wire by first receiving the waveform over the audio cable, saving the received transmission to a file, then processing the waveform. I did this incase I needed to make further adjustments to my code and I did not want to have to rerecord the audio.

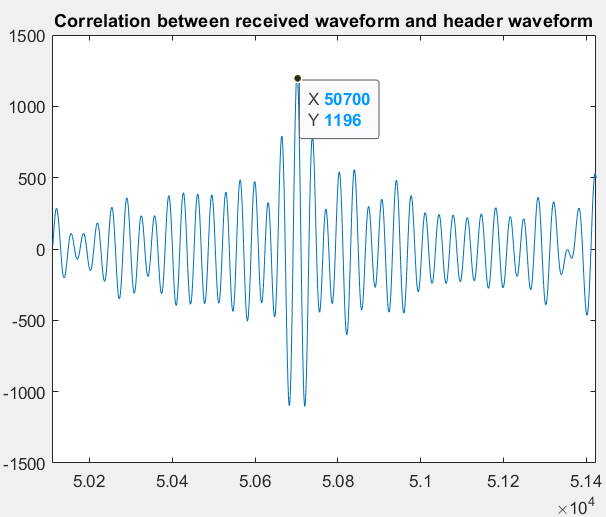
*Matlab Plots:*

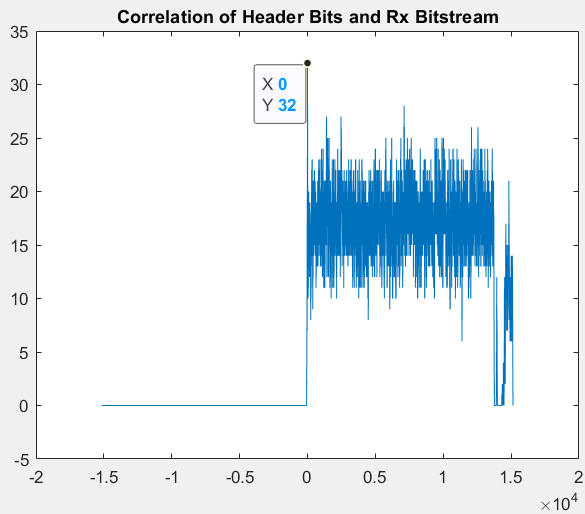












*Matlab Console Output (Over Wire):*

Header Sample Offset

50703

sampleOffset =

1

receivedText =

'ùÿ Who has believed our message and to whom has the arm of the Lord been revealed? 2 He grew up before him like a tender shoot, and like a root out of dry ground. He had no beauty or majesty to attract us to him, nothing in his appearance that we should desire him.3 He was despised and rejected by mankind, a man of suffering, and familiar with pain. Like one from whom people hide their faces he was despised, and we held him in low esteem. 4 Surely he took up our pain and bore our suffering, yet we considered him punished by God, stricken by him, and afflicted. 5 But he was pierced for our transgressions, he was crushed for our iniquities; the punishment that brought us peace was on him. and by his wounds we are healed. 6 We all, like sheep, have gone astray, each of us has turned to our own way; and the Lord has laid on him the iniquity of us all.  | ,-rÄà

aúX÷¿úr0q<»"ß‑1eHÿ'

As we can see from the Matlab console output of the received transmission, I was able to successfully receive the entire Isaiah passage error free. There is junk at the beginning and end because I do not truncate the header and trailer and everything after. However, the transmission is perfect.

## Over Air

I used the same code for the transmission over the air. I just simply set my audio manager to use the microphone instead, and my recordAudio() function automatically sets my input device to what the audio manager is set to.

*Matlab Main Script Code:*

% Christopher Hansen

% Communication Systems

% Lab 05

% 4/6/2021

%% Text to ASCII encoding then to binary bitstream

clear; clc;

text = fileread('Bible.txt');

%text = 'Christopher David Hansen';

bitstream = textToBitstream(text);

disp(bitsToText(bitstream));

bitstream = addHeaderTail(bitstream); % Add preamble and postamble

bitstream = convoEncode(bitstream);

%% MSK Modulation

Fs = 48000;

fc = 1200;

baud = 800;

T = 1/baud;

[s, t] = modulateMSK(bitstream, fc, baud, Fs);

figure();

plot(t, s);

title('4 Symbol Periods of MSK Modulated Source Signal');

xlabel('time (s)');

ylabel('Amplitude');

xlim([0 4\*T]);

hold off;

% Power Spectral Density

[Hs,f] = pwelch(s,[],[],[],Fs);

figure();

plot(f, pow2db(Hs));

title("Power Spectral Density of Source Signal");

xlabel("Hz");

ylabel("PSD (dB/Hz)");

%% Transmit

s = 0.99\*s;

audiowrite('IsaiahEncoded.wav', s, Fs); % write the generated sinsoid to .wav file

% with specified sampling frequency

%% Receive Waveform

[s\_rx,Fs] = audioread('rxIsaiahEncodedAir.wav'); % Read in generated waveform from .wav file

%s\_rx = recordAudio(20, Fs); % Receive the signal

s\_rx = s\_rx';

figure();

t\_tx = 0:1/Fs:(length(s\_rx)-1)/Fs;

plot(s\_rx);

title('Received Signal');

xlabel('Samples');

ylabel('Amplitude');

% Power Spectral Density

[Hs\_rx,f\_rx] = pwelch(s\_rx,[],[],[],Fs);

figure();

plot(f\_rx, pow2db(Hs\_rx));

title("Power Spectral Density of Received Signal");

xlabel("Hz");

ylabel("PSD (dB/Hz)");

%% Demodulation and Decoding

hexHeader = ['00'; 'FF'; 'FF'; '00']; % Header in hexadecimal

% ==== Following lines converts header to a wave ====

binHeader = [hexToBinaryVector(hexHeader(1,:),8)...

hexToBinaryVector(hexHeader(2,:),8)...

hexToBinaryVector(hexHeader(3,:),8)...

hexToBinaryVector(hexHeader(4,:),8)];

binHeaderEncoded = convoEncode(binHeader); % encode header bits, since they were added on as encoded bits at the start

s\_rx3 = deleteHeadTail2(binHeaderEncoded, s\_rx, fc, baud, Fs); % Sync Header Waveform

rxBitstream = demodMSK(s\_rx3, fc, baud, Fs); % MSK Demodulation

sampleOffset = syncBits(binHeaderEncoded, rxBitstream); % Sync the characters using header bits

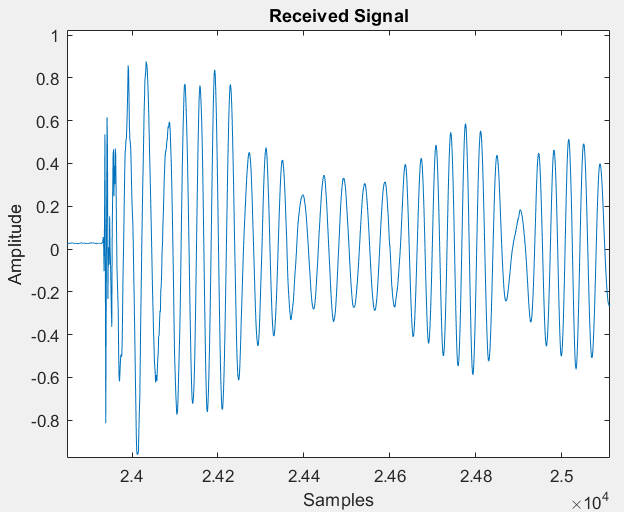
rxBitstream = convoDecode(rxBitstream); % Decode the bitstream

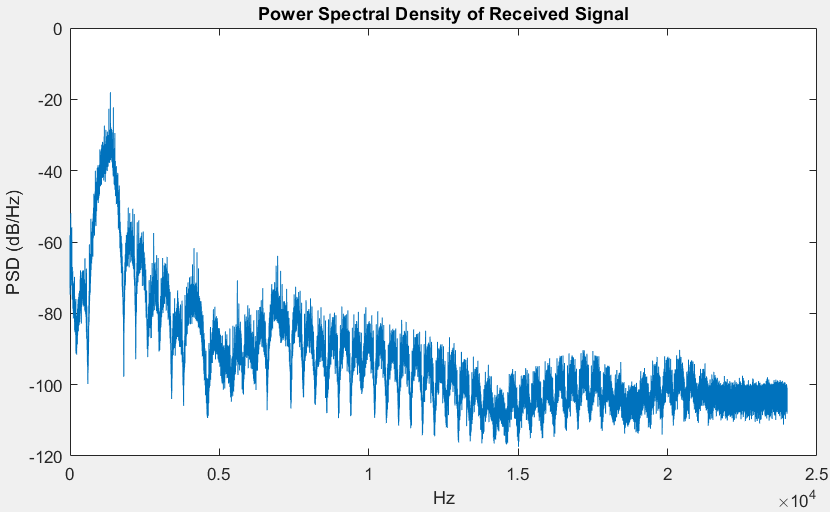
% = Convert to Text =

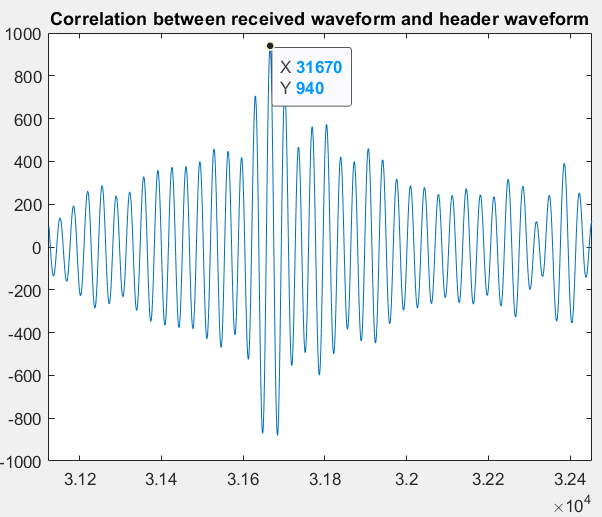
receivedText = bitsToText(rxBitstream(sampleOffset:8\*floor(length(rxBitstream)/8)))

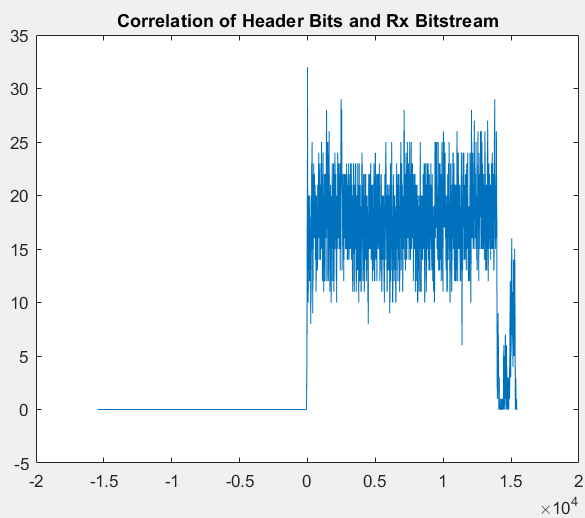
*Matlab Plots:*

I will not plot the source signal or source signal PSD, as they are identical to the plots from when I sent it over wire.









*Matlab Console Output:*

recording

done recording

>>

>> audiowrite('rxIsaiahEncodedAir.wav', s\_rx, Fs);

>>

Header Sample Offset

19345

sampleOffset =

1

receivedText =

'ùÿ Who has believed our message and to whom has the arm of the Lord been revealed? 2 He grew up before him like a tender shoot, and like a root out of dry ground. He had no beauty or majesty to attract us to him, nothing in his appearance that we should desire him.3 He was despised and rejected by mankind, a man of suffering, and familiar with pain. Like one from whom people hide their faces he was despised, and we held him in low esteem. 4 Surely he took up our pain and bore our suffering, yet we considered him punished by God, stricken by him, and afflicted. 5 But he was pierced for our transgressions, he was crushed for our iniquities; the punishment that brought us peace was on him. and by his wounds we are healed. 6 We all, like sheep, have gone astray, each of us has turned to our own way; and the Lord has laid on him the iniquity of us all.%ªÿ'

>>

The Matlab console output shows that we were able to receive the entire passage perfectly. We had perfect transmissions over file, wire, and air!