1 Introduction

In this project, you are going to implement the receiver for an acoustic modem. An acoustic modem is a device which sends digital signals using sound waves. This approach was used for internet access for many years.

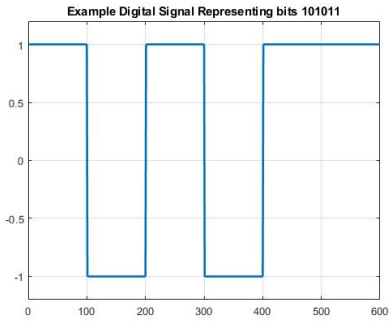
Nowadays it is still the method of communication of choice for underwater communications because acoustic signals propagate better than electromagnetic signals in salt water.

The goal is for you to decode a message contained in an acoustic signal.

2 The approach

The basic idea here is to use amplitude modulation to transmit signals through the acoustic channel.

Digital information can be transmitted using a sequence of pulses, with positive pulses representing ones and negative pulses representing zeros. For example, a continuous-time (CT) waveform $m(t)$ can be used to represent a bit sequence '101011', as shown below:



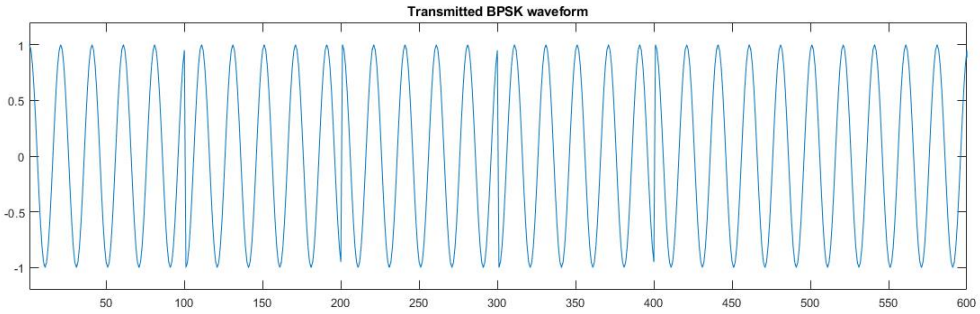
In this example, a pulse of width 100 time units is used. The time between the pulses is known as the Symbol Period.

The message signal $m(t)$ cannot be transmitted through the air. Instead, $m(t)$ will have to be translated to higher frequencies by multiplying it with a high-frequency cosine. This is the same approach used in Amplitude Modulation that you saw earlier in Signals.

Given the digital message signal is $m(t)$, then the transmitted signal is

$$x(t) = m(t)\cos(2\pi f_c t) \tag{1}$$

where f_c is known as the carrier frequency. The $x(t)$ signal would look something like the following:



This type of data modulation is called binary-phase-shift keying, because the information is carried in the phase of the cosine, and binary because there are one of 2 possible phase offsets to the cosine. Each transition from a 0 to a 1 and vice-versa results in a phase shift of the cosine.

Note that in practice, the frequency of the cosine should be considerably higher than what's illustrated in the graph, but we used a lower frequency cosine so the graph is easier to read.

The waveform $x(t)$ is what should be transmitted from the speaker of an acoustic modem.

At the receiver end, the received signal $y(t)$ is multiplied by a cosine and low-pass filtered, to produce an approximation to $m(t)$, which can be used to decode the transmitted bits. You'll find it helpful to refer to Homework 2 for how Amplitude Modulation (AM) works.

3 What you need to do

3.1 Minimum Viable Product

This project uses the following provided files:

- modem_tx.m - script to generate a transmit signal vector and store it into a .wav file. This is provided as a reference only. The signals have already been transmitted and recorded for you.
- modem_rx_starter.m - starter script to build your receiver
- short_modem_rx.mat and long_modem_rx.mat are two files containing received samples of a short and long transmission of data, respectively.
- find_start_of_signal.m - function used to detect when the signal is present in the received data
- BitsToString.m
- StringToBits.m

Your job is to decode the messages contained in the two .mat files. Start with the short file. The message in it is "Hello". The transmit code is given to you for reference in modem_tx.m

3.2 Possible directions for further exploration

1. Generate new data and transmit it over the air. Please see section on practical considerations below.

2. Reduce the symbol period. The symbol period directly impacts how quickly your data is transmitted. The smaller you can make it, the higher your data rate. (Note that you will have to do the previous part first).
3. Transmit Quadrature signals. You can transmit two independent bit-streams through the channel, one modulated by cosine and another modulated by sine and have the signals added together. To decode, you simply multiply by the corresponding cosine/sine signal and low-pass filter the result.
4. Use multi-level pulses, e.g., 4 levels of pulses with possible heights of -1, -1/3, 1/3, 1.

4 Practical Considerations

4.1 Lining up the signals

The data in the .mat files was collected as follows:

1. modem_tx.m was run
2. The resulting .wav file was transferred to a mobile phone
3. A recorder object was created in MATLAB and recording was started. You can see MATLAB documentation on how this is done.
4. The .wav file was played on the mobile phone about 2 feet away from the laptop which was recording on MATLAB.
5. The recorded data was stored in the vector y_r.

Since the receiver was started first (this is done to ensure that we don't miss the start of the transmission), the signal lives somewhere in the middle of y_r. You can plot y_r to see this. So we need to truncate it to start at the correct point. The standard way to do this is to prepend a known, but noise-like signal before the actual data bearing signal. Then an operation called cross correlation can be done to determine the start of the signal. This has already been done for you in the modem_tx.m and modem_rx_starter.m files, so you don't need to worry about this part.

4.2 Flipping signals

Due to propagation through the channel between the speaker on the phone and the mic on the computer, there is a good chance that the signal could arrive with its sign flipped at the receiver. You could flip it by hand or implement some automatic way to check if a sign flip is necessary. Note that in the two data files we provided for you, it is not necessary to flip the sign. You will only need to do this if you choose to make your own transmissions.

5 Deliverables

1. A brief report that includes an introduction, block diagram, equations used, graphs of signals in the time domain, and graphs of signals in the frequency domain.
2. Link to audio recordings if you produced new ones.
3. Your code.
4. Short demo video (2 minutes or less)