This lab will have you create and demodulate (in the presence of noise) a simple amplitude shift-keyed (ASK), frequency shift-keyed (FSK), and phase shift-keyed (PSK) waveform.

**Part 1: Pre-Lab Theory**

The performance of digital bandpass communication signals can be represented mathematically by:

**Amplitude Shift Keying (ASK):**

\[ P_e = Q \left( \sqrt{ \frac{E_b}{N_0} } \right) \quad E_b = \frac{A^2 T_b}{4} \]

**Frequency Shift Keying (FSK):**

\[ P_e = Q \left( \sqrt{ \frac{E_b}{N_0} } \right) \quad E_b = \frac{A^2 T_b}{2} \]

**Phase Shift Keying (PSK):**

\[ P_e = Q \left( \sqrt{ \frac{2E_b}{N_0} } \right) \quad E_b = \frac{A^2 T_b}{2} \]

For this lab, the “hardware” (Part 3) ASK and PSK signals will operate on a 200 kHz cosine carrier. For FSK, the two tones will be at 80 kHz and 210 kHz. The message signal will be a 50 kbps binary bitstream.

1. Complete the theoretical predictions in Table 1 for the ASK, FSK, and PSK waveforms.

**Table 1: Theoretical Characteristics of the Digital Bandpass Signals**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>ASK</th>
<th>PSK</th>
<th>FSK</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Energy Per Bit</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( P_e ) for ( \frac{E_b}{N_0} = 0 \text{ dB} )</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( P_e ) for ( \frac{E_b}{N_0} = 2 \text{ dB} )</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( P_e ) for ( \frac{E_b}{N_0} = 4 \text{ dB} )</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( P_e ) for ( \frac{E_b}{N_0} = 6 \text{ dB} )</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( P_e ) for ( \frac{E_b}{N_0} = 8 \text{ dB} )</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( P_e ) for ( \frac{E_b}{N_0} = 10 \text{ dB} )</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**Part 2: Simulation**

To evaluate the performance of ASK, FSK, and PSK operating in an AWGN channel, you will need to accurately simulate each modulation scheme and compare the results to the theoretical predictions. Simulating the performance of communication system is sometimes an arduous task, but it can basically be broken down into the following steps:

- **Generate a random sequence of bits (1’s and 0’s)**
- Input the bits to a modulator
- Add noise to the modulated signal based on a specific SNR value
- Demodulate and recover the bitstream
- Determine how many bits were received in error
- Calculate the bit error rate

**For this lab, the random bit sequence has been pre-generated by the instructor and is available on the course website in the file askfskpsk_511bits.txt. This bit sequence will be used for both simulation and hardware measurements. Knowing the bit sequence *a priori* is required in order to calculate the BER from the hardware data capture.**

To load the bit sequence into Matlab for your simulation, use the command:

```matlab
bits = load('askfskpsk_511bits.txt');
```

For this part of the lab, the simulation will be performed at complex baseband, in order to avoid unnecessary runtimes associated with simulating bandpass (carrier) waveforms. The necessary modulators, demodulators, and noise adder functions have been provided (see below for descriptions), and your job is to encapsulate them into a simulation. In order to compare the different modulation techniques, you will need to generate a plot that compares the theoretical $\frac{E_s}{N_0}$ vs BER curve with the simulated curve. To save space, plot all curves in the same figure, as illustrated in Figure 2. Run your simulation for $\frac{E_s}{N_0}$ that varies from 0 to 10 dB, to correspond with your calculations in Table 1.

**Note:** To see the lower BER’s, you will need more than 511 bits. The easiest way to do this is to concatenate several of the ‘bits’ vector prior to running them through your simulation.

For the simulation, you will connect Matlab software routines based on the following block diagram:

![Figure 1: Block diagram of a BER simulation of a digital modulation scheme operating in an AWGN channel.](image-url)
**Matlab Functions**

The following Matlab functions can be found on the course website:

- AddNoise.m
- PhaseMod.m
- PhaseDemod.m
- AmpMod.m
- AmpDemod.m
- FSK2Mod.m
- FSK2Demod.m

**AddNoise**(x, \(E_{B0}\text{perBit}, \text{BitsPerSymbol}\)) – This function adds noise to the vector x. The output SNR per bit of this vector is determined by the input variable \(E_{B0}\text{perBit}\). Note that \(E_{B0}\text{perBit}\) is linear not in dB.

**PhaseMod**(x, \(k\)) – This function uses an input vector of data bits x of length N to create an output vector of symbols of length \(N/k\) where \(k = \log_2(M)\) is the number of bits per symbol and M is the number of possible symbols. The symbols are complex baseband representations of M-ary phase modulated signals.

**PhaseDemod**(x, \(k\)) – This function creates a vector of bits of length \(Nk\) (where \(k = \log_2(M)\) is the number of bits per symbol) from a length N vector of phase modulated symbols. The assumed modulation is M-PSK where \(M = 2^k\).

**AmpMod**(x, \(k\)) – This function uses an input vector of data bits x of length N to create an output vector of symbols of length \(N/k\) where \(k = \log_2(M)\) is the number of bits per symbol and M is the number of possible symbols. The symbols are complex baseband representations of M-ary amplitude shift keying modulated signals.

**AmpDemod**(x, \(k\)) – This function creates a vector of bits of length \(Nk\) (where \(k = \log_2(M)\) is the number of bits per symbol) from a length N vector of ASK modulated symbols.

**FSK2Mod**(x) – This function uses an input vector of data bits x of length N to create an output vector of symbols of length N. The symbols are complex baseband representations of Binary Frequency Shift Keying modulated signals.

**FSK2Demod**(x, \(k\)) – This function creates a vector of bits of length N from a length N vector of 2-FSK modulated symbols.
Part 3: Working with Real Signals

For this section, the instructors have captured a 2-ASK, 2-FSK, and BPSK signal with various levels of noise. Due to the difficulty in achieving and maintaining synchronization (required for the ideal matched filter receiver), you will be implementing an incoherent (“ye old threshold”) ASK and FSK demodulator and calculating the Bit Error Rate.

The noisy captures are available on Blackboard, as ASK_Noise_511.mat, FSK_Noise_511.mat, and PSK_Noise_511.mat. These were recorded using the askfskpsk_511bits sequence as the information signal. When you load these files, your workspace will be populated with the following three variables:

- \( R_b \) – This is the bit rate used to generate the ASK/FSK waveform (bits/sec).
- \( f_s \) – This is the sampling rate used to capture the ASK/FSK waveform (Hz).
- \( s_{\text{noise}} \) – This is a matrix of the noisy ASK/FSK signals corresponding to the \( E_b/\text{N}_0 \) values in the theory/simulation (0 to 10 dB in steps of 2 dB). The matrix row corresponds to a particular \( E_b/\text{N}_0 \) values, and the columns are the recorded samples. For example, \( s_{\text{noise}}(1,:) \) is the capture of an ASK/FSK signal at 0 dB \( E_b/\text{N}_0 \), \( s_{\text{noise}}(2,:) \) is the capture of an ASK/FSK signal at 2 dB \( E_b/\text{N}_0 \), etc.

\[ P_e = \frac{1}{2} e^{-\frac{E_b}{2\text{N}_0}} \]

\[ \Rightarrow \] You will need two more Matlab functions from the website, filter_digital.m and filter_BB.m. These filters behave exactly the same way as filter_RF.m, filter_IF.m, or filter_baseband.m, but have a constant group delay (linear phase) which makes it cleaner to implement your receiver.
**Part 3a: ASK with Random Bits**

1. Load the file `ASK_Noise_511.mat` into your Matlab workspace (you can do this either in the command window or as part of an m-file). Verify that all three variables are present and accessible.

2. Using the block diagram in Figure 3 as a guide, create an incoherent receiver in Matlab to demodulate the ASK Waveform through the envelope detector, $z(t)$. Plot and visually verify that the waveform out of the envelope detector is a noisy version of a baseband digital line code.

   ![Block diagram of a simple incoherent 2-ASK receiver.](image)

   **Figure 3:** Block diagram of a simple incoherent 2-ASK receiver.

3. Next, recall that the incoherent receiver is essentially a “sample and hold” receiver. You will need to establish a point on each bit to use as the sampling instance for your demodulator, similar to what is shown in Figure 4 below. Ideally, this would be at the midpoint of each bit, which you can find by calculating the number of samples per bit. Note that the filter will add a slight delay to the received signal, so the output bit sequence contained in $z(t)$ will **NOT** start at $time = 0$.

   ![Illustration of “sample and hold” process for the incoherent receiver.](image)

   **Figure 4:** Illustration of “sample and hold” process for the incoherent receiver.

4. You will need to establish a threshold to compare your sample to in order to determine whether the received bit was a 1 or a 0. Ideally, the decision threshold will be halfway between the maximum and minimum value of the output of the envelope detector, $z(t)$. You may, however, need to tweak this value slightly in order to better match the theoretical Probability of Error curve.

5. Finally, compute the BER of your receiver and compare it against the theoretical curve. Note that you may have to adjust the timing and threshold values, but you should be able to get within 1 dB or so of the theoretical curve, as illustrated in Figure 5 below:
Part 3b: FSK with Random Bits

1. Load the file FSK_Noise_511.mat into your Matlab workspace (you can do this either in the command window or as part of an m-file). Verify that all three variables are present and accessible.

2. Using the block diagram in Figure 6 as a guide, create an incoherent receiver in Matlab to demodulate the FSK Waveform through the envelope detector, \( z_0(t) \) and \( z_1(t) \). Plot and visually verify that the waveform out of the envelope detector is a noisy version of a baseband digital line code and that the two waveforms are opposites.

Figure 6: Block diagram of a simple incoherent 2-FSK receiver.
3. Next, recall that the incoherent receiver is essentially a “sample and hold” receiver. You will need to establish a point on each bit at the output of the envelope detector combining circuit to use as the sampling instance for your demodulator, similar to what is shown in Figure 3 (above). Ideally, this would be at the midpoint of each bit, which you can find by calculating the number of samples per bit. Note that the filter will add a slight delay to the received signal, so the output bit sequence contained in the output of the envelope detector combined output will NOT start at \( t = 0 \).

4. You will need to establish a threshold to compare your sample to in order to determine whether the received bit was a 1 or a 0. Ideally, the decision threshold will be halfway between the maximum and minimum value of the output of the envelope detector (theoretically 0 for an FSK waveform). You may, however, need to tweak this value slightly in order to better match the theoretical Probability of Error curve.

5. Finally, compute the BER of your receiver and compare it against the theoretical curve. Note that you may have to adjust the timing and threshold values, but you should be able to get within 1 dB or so of the theoretical curve, as illustrated in Figure 7 below:

---

![Theoretical and Simulated BER for FSK operating in an AWGN Channel](image)

**Figure 7:** Example Theoretical and Simulated BER curves for 2-FSK Modulation.
Part 3c: PSK with Random Bits

1. Load the file `PSK_Noise_511.mat` into your Matlab workspace (you can do this either in the command window or as part of an m-file). Verify that all three variables are present and accessible.

2. Using the block diagram in Figure 8 as a guide, create a coherent receiver in Matlab to demodulate the PSK waveform through the lowpass filter, $z(t)$.

   **Note:** To ease the burden of synchronization and avoid filter issues, the AWGN was pre-filtered and then added to the PSK signal. Thus, you do not need to add a bandpass filter in your receiver (filtering the noise and then adding it has the same effect as adding the noise and filtering).

   ![Block diagram of a PSK receiver](image)

   **Figure 8: Block diagram of a PSK receiver.**

3. Next, recall that the coherent receiver is essentially a “integrate and dump” receiver. As with the ASK and FSK receivers, the filter will add a slight time delay to the received signal, so output bit sequence will **NOT** start at time $t=0$. You will need to remove these samples prior to integrating. To perform the integration, use the Matlab command `intdump(z,sam_per_bit)`, where $z$ is the output of the correlator and `sam_per_bit` is the number of samples per bit.

4. You will need to establish a threshold to compare your sample to in order to determine whether the received bit was a 1 or a 0. Ideally, the decision threshold will be halfway between the maximum and minimum value of the output of the integrator. You may, however, need to tweak this value slightly in order to better match the theoretical Probability of Error curve.

5. Finally, compute the BER of your receiver and compare it against the theoretical curve. Note that you may have to adjust the timing and threshold values, but you should be able to get within 1 dB or so of the theoretical curve, as illustrated in Figure 9 below:
Figure 9: Example Theoretical and Simulated BER curves for BPSK.
Part 4: Impact of BER on Audio Quality

In the world of Analog Communications, AWGN has a direct impact on the SNR of AM and FM communications, which in turn had a direct impact on the demodulated audio quality (Subjective Quality Factor). In digital communications, the BER will have a direct impact on SQF, although the improvement is much more dramatic.

To illustrate these effects, a brief snippet of a popular song was digitized, ASK/FSK modulated at $\frac{E_b}{N_0}$’s of 10, 12, and 14 dB. The modulated data are available on a Google Drive as ASK_Song_Noise.mat and FSK_Song_Noise.mat.

1. Load the file ASK_Song_Noise.mat into your Matlab workspace (you can do this either in the command window or as part of an m-file). Verify that all three variables are present and accessible.

2. Demodulate the signal using your ASK receiver from Part 1 and compute/plot the BER against the theoretical curve. You may have to adjust your threshold parameter slightly, but should still be able to come within 1 dB or so of the theoretical curve.

3. To recover and play the original audio signal, you will need to execute the following Matlab commands:

   ```
   audio_DAC = Digital2Analog(demod_bits,8);
   soundsc(audio_DAC,8000,8)
   ```

4. Evaluate the sound quality for the three different $\frac{E_b}{N_0}$ and BER’s. Repeat for FSK and compare with ASK. Note that, theoretically, the two should perform identically, however the instructor did achieve slightly better performance with FSK.