LAB 9

Binary Phase-Shift Keying (BPSK)

Prerequisites: Lab 5 – Double-Sideband Suppressed Carrier, Lab 7 – Amplitude-Shift Keying

9.1 Objective

In phase-shift keying (PSK), information is encoded on the phase of the transmitted carrier, rather than on its amplitude (ASK) or its frequency (FSK). In binary phase-shift keying (BPSK) there are two phase values, $0°$ and $180°$, which means that an unmodified carrier is transmitted to represent one binary data value, while an inverted carrier is transmitted to represent the other binary data value.

BPSK is the digital version of double-sideband suppressed-carrier analog modulation. We will see that the BPSK transmitter and receiver have a DSB-SC transmitter and receiver at their core. The additional features such as symbol mapping, pulse shaping, matched filtering, threshold detection, and pulse and frame synchronization that are needed in ASK and FSK systems are also needed in PSK systems. Just as in DSB-SC, the absence of a transmitted carrier leads to a reduction in power needed for a given level of performance. BPSK is optimum among binary systems in providing the lowest average power needed for a given bit error rate. BPSK is easily extended to quadrature phase-shift keying (QPSK) and quadrature amplitude modulation (QAM), as we shall see in a subsequent lab exercise. These extensions allow transmission of multiple bits per pulse and can be very effective for transmitting data at high rates over a channel of limited bandwidth.

Demodulating a BPSK signal requires synchronizing the phase of the received signal. Phase synchronization was also encountered in the DSB-SC lab project. We will see that the difficulties of phase synchronization can be avoided, at a small performance penalty, by using differential encoding of the transmitted data. Differential binary phase-shift keying (DPSK) is a robust and efficient modulation method that is widely used in practice.

9.2 Background

The generation and detection of an analog DSB-SC signal is discussed at length in the background section of *Lab 5: Double-Sideband Suppressed-Carrier*. It is strongly suggested that this material be reviewed at this point.

Transmitter

A binary phase-shift keyed signal is a train of pulses, each of the form

$$Ag_{TX}(t)\cos(2\pi f_c t + \theta). \quad (1)$$

In Eq. (1), A is a constant that sets the transmitted power level, $g_{TX}(t)$ is a fixed pulse shape, f_c is the carrier frequency, and θ takes a value of either $0°$ or $180°$ to carry the desired information. Note that we can also write Eq. (1) as

$$\pm Ag_{TX}(t)\cos(2\pi f_c t), \quad (2)$$

where the plus sign corresponds to $\theta = 0°$ and the minus sign to $\theta = 180°$. We will assume, as in the previous digital signaling lab projects, that a new pulse is transmitted every T seconds, so that the symbol rate is $1/T$ symbols/s. For a binary scheme such as BPSK, the bit rate is the same as the symbol rate. Since the pulse $g_{TX}(t)$ does not carry information, its shape can be chosen to satisfy other criteria. As was the case with ASK, we desire a pulse shape that provides a rapid spectral rolloff and minimizes intersymbol interference. We will use the "root-raised-cosine" pulse shape, as we did in the ASK lab.

The steps needed to form a BPSK signal should be familiar if you have completed the ASK lab project:

1. Symbol Mapping. The input data arrives as a stream of bits. Recall that the *MT Generate Bits* function produces an array of bytes containing the numbers 1 and 0. In the symbol mapping step, the bits are replaced by numerical values. For BPSK we will represent a binary 1 by the complex double $1 + j0$ and a binary 0 by the complex double $-1 + j0$. Note that we are representing bits by complex numbers, even though the imaginary parts are zero. This is because the USRP requires a complex-valued input, and because in a future lab we will use the imaginary part to carry additional data. Table 1 shows the BPSK symbol mapping.

Table 1. BPSK Symbol Mapping

Bit Value	Symbol

0	$-1+j0$
1	$1+j0$

2. Upsampling. As a first step toward replacing symbols with pulses, we will place $L-1$ zeros after each symbol. This produces a sample interval of

$$T_x = \frac{T}{L}, \tag{3}$$

or a sample rate of

$$\frac{1}{T_x} = L\frac{1}{T}. \tag{4}$$

A higher upsampling factor L makes the D/A conversion in the transmitter easier, but requires faster digital processing.

3. Pulse Shaping. If the upsampled signal is applied to a filter whose impulse response $g_{TX}[n]$ is a root-raised-cosine pulse, then each symbol at the filter output will be represented by a root-raised-cosine pulse. Steps 1 through 3 convert the input bit stream to a polar message signal. Note that pulse shapes other than root-raised-cosine can be used simply by changing the shape of the impulse response $g_{TX}[n]$. The root-raised-cosine pulse has a very rapid spectral rolloff, so that the transmitted signal will not cause interference to signals at nearby carrier frequencies.

4. Modulation. The polar message signal, consisting of a train of pulses of the form $(\pm 1+j0)g_{TX}[n]$, can be sent directly to the USRP transmitter. The USRP will convert the signal to continuous time and add the carrier as shown in Eq. (2).

Receiver

A PSK receiver begins with a DSB-SC demodulator. When the transmitted BPSK signal arrives at the receiver it has the form of a train of pulses, each given by

$$r(t) = \pm D g_{TX}(t) \cos(2\pi f_c t + \phi), \tag{5}$$

where D is a constant (usually much smaller than the constant A in the transmitted signal) and the angle ϕ represents the difference in phase between the transmitter and receiver carrier oscillators. If the receiver's carrier oscillator is set to the same frequency as the transmitter's carrier oscillator, the USRP receiver will do most of the work in demodulating the BPSK signal. The receiver's *Fetch Rx Data* will provide a train of output pulses, each given by

$$\tilde{r}[n] = \pm \frac{D}{2} g_{TX}[n] e^{j\phi}. \tag{6}$$

The sampling rate in Eq. (6), $1/T_z$, is set by the receiver's "IQ rate" parameter. This rate is set to provide M samples every T seconds, where $1/T$ is the symbol rate.

We can remove the phase offset ϕ using any one of a variety of techniques. The technique used in the DSB-SC lab project works well. To summarize, we begin by squaring $\tilde{r}[n]$, giving

$$\tilde{r}^2[n] = \frac{D^2}{4} g_{TX}^2[n] e^{j2\phi}. \tag{7}$$

This step eliminates phase changes caused by the data, as well as phase changes caused by changes in the polarity of $g_{TX}[n]$. Next, the angle 2ϕ can be extracted using a *Complex to Polar function* from the Data Types→Numeric→Complex palette. It turns out to be helpful at this point to smooth variations in 2ϕ caused by noise. The *median filter* (Analysis→Signal Processing→Filters→Special Filters) does a good job. The default values can be accepted for the "left rank" and "right rank" parameters. Next the *Unwrap Phase* from the Analysis→Signal Processing→Conditioning palette will remove jumps of $\pm 2\pi$. Finally, dividing by two gives the desired estimate of the phase error ϕ. The block diagram in Figure 1 shows the entire phase synchronization process.

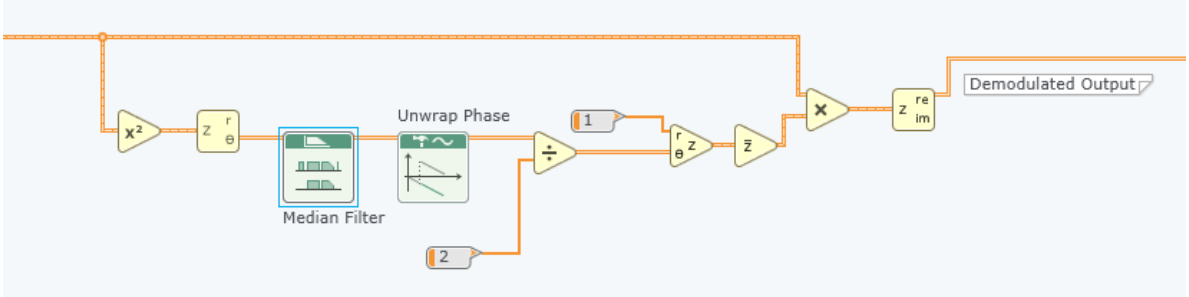

Figure 1. Carrier Phase Synchronization

The remaining steps carried out by the receiver should be familiar from the ASK and FSK labs. These steps are:

1. Matched Filtering. We will use a root-raised-cosine receiver filter. This filter's impulse response $g_{RX}[n]$ is matched to the pulse shape $g_{TX}[n]$ of the received pulses. The matched filter gives optimum performance in the presence of additive, white, Gaussian noise. Further, the cascade of the two root-raised-cosine filters $g_{TX}[n]$ and $g_{RX}[n]$ produce a raised-cosine pulse shape that is free from intersymbol interference.

2. Pulse Synchronization. The matched filter output is an analog baseband signal that must be sampled once per symbol time, i.e. once every T seconds. Because of filtering, propagation delays, and distortion caused by the communication channel, it is necessary to determine the optimum time to take these samples. A function called *PulseAlign(real)* has been provided to align the baseband signal so that the sample at index 0 is the correct first sample.

3. Sampling. The *Decimate* function will sample the aligned baseband waveform at index 0 and every T seconds thereafter.

4. Detection. Once the baseband waveform has been sampled, each sample must be examined to determine whether it represents a symbol of value 1 or a symbol of value 0.

5. Symbol Mapping. The detected symbol values must be converted to bits. For binary PSK, this step is easily included in the detection step.

9.3 Pre-Lab

Transmitter

1. Create a program to generate a BPSK signal using the USRP. A template for the transmitter has been provided in the file *BPSKTxTemplate.gvi*. This template contains the four functions for interfacing with the USRP along with *MT Generate Bits* from the Modulation Toolkit. *MT Generate Bits* will create a pseudorandom sequence of bits that can serve as a data sequence for testing your BPSK system. Note that by default, *MT Generate Bits* will produce the same sequence of bits every time you run the program. This is useful for debugging, but if you would like to generate a different sequence of bits every time, wire a random number to the "seed in" input. The steps below contain details about how to create the required transmitter.

2. First do the symbol mapping, as shown in Table 1.

3. Upsample the array of symbols using *Upsample* from the Analysis→Signal Processing→Conditioning subpalette. In this lab project you are given control inputs to set the symbol rate $1/T$ and the IQ rate $1/T_x$. Set the symbol rate to 10,000 symbols/s and the IQ rate to 200×10^3 Sa/s. Use the symbol rate and coerced IQ rate to calculate the upsampling factor L.

4. Use *MT Generate Filter Coefficients* from the Modulation Toolkit to generate the pulse shaping filter. (*MT Generate Filter Coefficients* can be found on the Analysis→ Communications→Digital→Utilities subpalette.) Set the modulation type to PSK, and the pulse-shaping filter "samples per symbol" to your calculated value of L. Create a front-panel control for "pulse shaping filter" and set this to "Root Raised." Wire the "pulse shaping filter coefficients" output to the "Y" input of a *Convolution*. The *Convolution* is available from the Analysis→Signal Processing→Operation subpalette. Wire the output from your upsampler to the "X" input of the *Convolution*.

5. Normalize the amplitude of your filtered message signal to a maximum absolute value of 1. The *Quick Scale 1D function* in the ExternalFiles folder will find the maximum of the absolute value. To ensure that your scaled message remains complex-valued, use a separate division function to do the actual scaling. Connect the scaled message to the *Build Waveform* function that connects to the Baseband Waveform graph provided in the template. Also send the scaled message to the *Write Tx Data* function.

6. To observe the spectrum of the transmitted signal, wire the complex baseband waveform to the *Power Spectrum for 1 Chan(CDB)* that connects to the Power Spectrum graph provided in the template.

This completes construction of the BPSK transmitter. Save your transmitter in a file whose name includes the letters "BPSKTx" and your initials (e.g. *BPSKTx_BAB.gvi*).

Receiver

1. Create a program to implement a BPSK receiver using the USRP. A template for the receiver has been provided in the file *BPSKRxTemplate.gvi*. This template contains the six interface functions for interfacing with the USRP.

   Calculate the "number of samples" for the *Fetch Rx Data* function to fetch using the message length, symbol rate front panel inputs and the coerced IQ rate. Double the number of samples the receiver will fetch to acquire two frames of data. Since the receiver's starting point is random, this ensures that there will be one complete frame of received data in the block of samples fetched.

   Implement the carrier phase synchronization as shown in Figure 1.

2. To implement the receiver's matched filter, use *MT Generate Filter Coefficients* just as you did for the transmitter. Set the modulation type to PSK, and calculate the "matched samples per symbol" M from the "actual IQ rate ($1/T_z$)" and the symbol rate ($1/T$) obtained from the front-panel control. Create a front-panel control for "pulse shaping filter" and set this to "Root Raised." Wire the "matched filter coefficients" output to the "Y" input of a *Convolution*. The output of your pulse shaping filter should be connected to the

Cluster Properties function provided in the template. The *Cluster Properties function* feeds the Baseband Output graph.

3. Place the *PulseAlign (real)* function from the ExternalFiles folder on your block diagram and wire the baseband output waveform to the "input waveform" input and wire the M samples/symbol to the "receiver sampling factor" input.

   Once the baseband waveform is aligned, it can be sampled. The *Decimate (single shot)* function can be obtained from the Analysis→Signal Processing→Conditioning subpalette. The "decimating factor" is M .

4. The Modulation Toolkit function *MT Format Eye Diagram* has been provided in the receiver template. Wire the baseband output waveform to the "waveform" eye-diagram input. The "symbol rate (Hz)" input value is available from the front panel control. Set the "eye length" parameter to 2.

5. To determine whether each received sample is more likely to represent a 1 or a 0, the sample must be compared with a threshold. Because the message is a polar signal, the threshold can be taken as zero. The result of this comparison is the receiver's digital output. The output of the comparison will be a Boolean array. You can convert this array to an integer array by using a *Boolean To Integer function*.

This completes construction of the BPSK receiver. Save your receiver in a file whose name includes the letters "BPSKRx" and your initials (e.g. *BPSKRx_BAB.gvi*).

Questions

1. In *Transmitter* Step 3 you are given $1/T = 10{,}000 \text{ symbols/s}$ and $1/T_x = 200 \times 10^3 \text{ Sa/s}$. Find the corresponding value for the number of samples per symbol L .

2. Eq. (6) in the Background section above shows that the received baseband signal $\tilde{r}[n]$ includes a factor $e^{j\phi}$, where ϕ is any phase difference that may exist between the transmitter and receiver carrier oscillators. Explain what would happen if you omitted the

phase synchronization step in the receiver. Specifically, what would be the receiver output if ϕ just happened to take the value $\pi/2$?

3. In *Receiver* Step 2 the "actual IQ rate" $1/T_z$ may be different from the rate $1/T_x$ that was used at the transmitter. (Note that the symbol rate $1/T$ must be the same at the transmitter and receiver.) The value of the receiver's IQ rate determines the receiver sampling factor M. What is the advantage to using a higher value of M? What is the advantage of using a lower value of M?

4. Although BPSK is a suppressed-carrier version of ASK, we do not use an envelope detector to demodulate BPSK the way we did for ASK. Why is that? What would the receiver output be if we used an envelope detector for demodulation?

5. Bit error rate is a measure of performance of a digital communication system. What would the bit error rate be if the transmitter failed and the receiver received only noise? Explain your reasoning.

9.4 Lab Procedure

1. Connect a loopback cable and attenuator between the TX 1 and RX 2 connectors of the USRP. Connect the USRP to your computer and plug in the power to the USRP. Run LabVIEW and open the transmitter that you created in the prelab.

2. Ensure that the transmitter is set up to use

   Carrier Frequency: 915.0 MHz

   IQ Rate: 200 kHz. Note: This sets the value of $1/T_x$.

   Gain: 0 dB.

   Active Antenna: TX1

   Symbol rate: 10,000 symbols/s

   Message Length: 1000 bits

   Pulse shaping filter: Root Raised

Run the transmitter. Use the large STOP button on the front panel to stop transmission connectors.

3. After running the transmitter, observe the spectrum of the transmitted signal. Measure the "main lobe" bandwidth of the transmitted signal. Change the pulse shaping filter control to "none" to create rectangular pulses and run the transmitter again. Compare the spectrum of the transmitted signal with the spectrum for root-raised-cosine pulses. Return the pulse shaping filter control setting to "Root Raised."

4. Ensure that the receiver is set up to use

   Carrier Frequency: 915.0 MHz

   IQ Rate: 400 kHz. Note: This sets the value of $1/T_z$.

   Gain: 0 dB

   Active Antenna: RX2

   Symbol rate: 10,000 symbols/s

   Message Length: 1000 bits

   Pulse shaping filter: Root Raised

5. Run the transmitter, then run the receiver. Once the receiver has acquired a block of data, you may stop the transmitter.

6. Observe the eye diagram. Make note of the optimum sampling time and the presence or absence of intersymbol interference. To see the effect of pulse synchronization, move the waveform input of *MT Format Eye Diagram* to the "aligned waveform" output of *PulseAlign(real)*. Run the transmitter and receiver again. Observe the eye diagram. What is the optimum sampling time now?

7. Modify the transmitter to include the *AddFrameHeader(real)*, available in the *BasicUSRPLabs folder*. Place *AddFrameHeader(real)* after the symbol mapping, but before conversion of the symbols to complex. Next, modify the receiver to include the *FrameSync(real)*, also available from the *BasicUSRPLabs folder*. Place *FrameSync(real)*

immediately following *Decimate*. Wire the output of *Decimate* to the "Sampled Input" of *FrameSync(real)*. Leave the remaining inputs of *FrameSync(real)* unwired. Wire the "Aligned Samples" output of *FrameSync(real)* to the threshold comparison function.

Wire the array of output bits from the threshold comparison to the "array" input of an *Array Subset function*. Set the "index" input to zero, and set the "length" input to the length specified by the "message length" control. Display the output of *Array Subset function* as "Output bits" on the receiver front panel.

Note that the "Output Signal" and "max index" outputs of *FrameSync(real)* will not be used in this lab project.

8. Automate measurement of the bit error rate (BER) by using the *MT Calculate BER* from the Modulation Toolkit (Analysis→ Communications→Digital→Measurements subpalette). From the Configure ribbon, choose "PN Fibonacci." Set the "BER trigger threshold" to 0.4. Connect indicators to the "BER" and "trigger found?" outputs. When you run the program, "trigger found?" will be true whenever the measured BER is below the BER trigger threshold.

   Run the transmitter and receiver. If everything is working correctly, the BER should be 0 or 1.0. Run the transmitter and then run the receiver a dozen times or so. Verify that about half the time the BER is 0 and about half the time the BER is 1.0.

Questions

1. How do the main lobe bandwidth and spectral rolloff rate for root-raised cosine pulses compare with the same quantities when rectangular pulses are used?

2. What does a BER of 1.0 signify? Explain why the BER is 0 half the time and 1.0 half the time.

Differential Encoding

Suppose the data bits are passed through the logic circuit shown in Figure 2 before being sent to the transmitter. In this circuit, the exclusive-nor gate will produce a 0 whenever the current

input differs from the previous output, and a 1 whenever the current input is the same as the previous output.

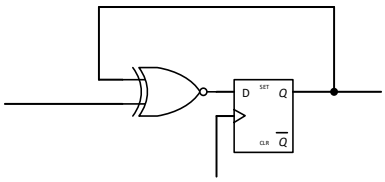

Figure 2. Differential Encoder

For example, an input of

$$1 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 0 \quad 1 \quad \ldots$$

produces an output of

$$0 \quad 0 \quad 1 \quad 1 \quad 1 \quad 0 \quad 1 \quad 0 \quad 0 \quad \ldots,$$

assuming that the flip-flop has an initial state of 0. Now suppose that the received bit sequence is passed through the circuit shown in Figure 3. The output will be a 1 whenever the current

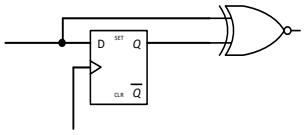

Figure 3. Differential Decoder

input and the previous input are the same, and the output will be a 0 whenever the current input and the previous input are different. For example, a received bit sequence of

$$0 \quad 0 \quad 1 \quad 1 \quad 1 \quad 0 \quad 1 \quad 0 \quad 0 \quad \ldots$$

produces an output of

$$1 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 0 \quad 1 \quad \ldots.$$

We see that the decoder of Figure 3 reverses the effect of the encoder of Figure 2. The important part of this is to notice what happens if the received bit pattern is inverted. If the received bit sequence is

$$1 \quad 1 \quad 0 \quad 0 \quad 0 \quad 1 \quad 0 \quad 1 \quad 1 \quad \ldots,$$

"same" and "different" are not altered, and so the decoder output will still be

$$1 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 0 \quad 1 \quad \ldots.$$

9. Add a differential encoder to your BPSK transmitter and a differential decoder to your receiver. Place the encoder immediately after *MT Generate Bits*, before the symbol mapping. Place the decoder after the threshold. Note that your transmitted sequence will have to be made one bit longer to include the initial state of the encoder flip-flop.

   Hint: The D flip-flop creates a one-sample delay. The delay is easily implemented in LabVIEW using a *Feedback Node function* from the Programming Flow palette.

10. Run the transmitter and then run the receiver a dozen times or so. You should find that the BER is always zero.

Differential Encoding, Part 2

Differential encoding can be done after the symbol mapping.

Table 2 is a truth table showing the correspondence between the exclusive-nor operation and ordinary multiplication. We will see below that the differential encoding allows the phase synchronizer to be eliminated from the receiver circuit.

Table 2. Exclusive Nor and Numerical Multiplication

Boolean				**Numerical**		
Input		Output		Input		Output
0	0	1		–1	–1	1
0	1	0		–1	1	–1
1	0	0		1	–1	–1
1	1	1		1	1	1

11. Implement the differential encoder numerically. Place the encoder in your transmitter following *AddFrameHeader(real)*. (This time the frame header also gets encoded.) Do not forget to add the initial state of the encoder to the beginning of the transmitted sequence.

Save your transmitter in a file whose name includes the letters "DPSKTx" and your initials (e.g. *DPSKTx_BAB.gvi*).

12. In the receiver, remove the phase synchronizer and run the received data directly to the *Convolution* that implements the matched filter. Place the differential decoder immediately after *Decimate*, the receiver's sampler, and before *FrameSync(real)*. Since the received data are complex-valued at this point, design your decoder to form the product of the current sample and the *complex conjugate* of the previous sample. Then take the real part of the result. You may also need to replace *PulseAlign(real)* with *PulseAlign(Complex).*

    Save your receiver in a file whose name includes the letters "DPSKRx" and your initials (e.g. *DPSKRx_BAB.gvi*).

12. Run the transmitter and then run the receiver several times. Verify that the BER is always zero.

Questions

1. Show that the differential encoder of Figure 2 and the differential decoder of Figure 3 continue to work properly as a system if the encoder's flip-flop has an initial state of 1.

2. Show, starting with Eq. (6), why the phase synchronizer is not needed in the DPSK receiver.

3. In the BPSK system, the eye diagram has the same amplitude every time you run the receiver. In the DPSK system, the amplitude of the eye diagram changes on every run. Explain why this happens. Explain whether, if noise were present, the BER would also change every time the DPSK receiver is run.

7. Differential phase-shift keying is intended to be robust in the presence of a phase difference between the transmitter and receiver oscillators. It is also possible for there to be a frequency difference between the oscillators. Suppose there is a small frequency difference of Δf between the transmitter and receiver oscillators. Modify Eq. (6) to take this frequency difference into account. How will the output of the DPSK receiver change

because of this frequency difference? Using the parameter values of this lab project, determine how large a frequency difference will be "significant." Be sure to specify what you mean by "significant."

9.5 Report

Prelab

Hand in documentation for the programs you created for the transmitter and receiver. Also include documentation for any functions you created. To obtain documentation, print out legible screenshots of the front panel and block diagram.

Answer all of the questions in the Prelab section marked *Questions*.

Lab

Submit the program you created for the transmitter and receiver. Include both the BPSK and DPSK programs. Also submit any functions you created. Be sure your files adhere to the naming convention described in the instructions above.

Submit documentation for the *DPSKTx* and *DPSKRx* programs. Resubmit documentation for any functions you modified during the lab.

Answer all of the questions in each of the Lab Procedure sections marked *Questions* above.