LAB 12

Quadrature Phase-Shift Keying

12.1 Objective

All of the modulation systems we have considered so far encode one bit of data into one symbol. For applications in which higher data rates are desired but channel bandwidth is limited, it is desirable to encode multiple bits of data into each symbol. In this lab project we introduce quadrature phase-shift keying (QPSK), a variation on binary phase-shift keying that encodes two bits of data into each symbol. Using QPSK, we can transmit data at twice the rate, without increasing the channel bandwidth. There will be some loss in performance, however, as somewhat more transmitted power will be needed to maintain a given bit error rate in the presence of noise.

The new concepts that are introduced in this lab project are the symbol mapping, which is more elaborate than the simple mapping used in BPSK, and the signal constellation, which is a visualization tool that is very useful for understanding high-efficiency modulation methods.

QPSK is easily extended to modulation methods that transmit even more than two bits per symbol. Some of these methods are known as "quadrature amplitude modulation" (QAM). For example, 16-QAM can transmit four bits of data per symbol. The more bits that are carried on each symbol, however, the more transmitted power will be needed to maintain a given bit error rate.

12.2 Background

As we saw in the BPSK lab project, a PSK signal can be represented as a train of pulses of the form

$$x(t) = Ag_{TX}(t)\cos(2\pi f_c t + \theta), \tag{1}$$

where A is a constant, $g_{TX}(t)$ is a fixed pulse shape designed to limit the signal bandwidth and control ISI, and f_c is the carrier frequency. Information is carried on the phase angle θ. For BPSK, θ can take one of two values, and each pulse carries one bit of information. For QPSK, θ can take four values. Since two bits are needed to specify one out of four possibilities, each QPSK pulse carries two bits of information. We will take the four possible phase angles to be $\theta = \pm\pi/4, \pm 3\pi/4$.

An alternative way of representing the pulse of Eq. (1) results from expanding the cosine. We can write

$$x(t) = A\cos(\theta)\, g_{TX}(t)\cos(2\pi f_c t) - A\sin(\theta)\, g_{TX}(t)\sin(2\pi f_c t). \tag{2}$$

Substituting the four possible values of θ gives

$$x(t) = \pm\left(A/\sqrt{2}\right) g_{TX}(t)\cos(2\pi f_c t) \mp \left(A/\sqrt{2}\right) g_{TX}(t)\sin(2\pi f_c t). \tag{3}$$

We can interpret Eq. (3) to imply that one data bit determines the polarity of the $\left(A/\sqrt{2}\right)g_{TX}(t)\cos(2\pi f_c t)$ term (the *in-phase* term), while the second data bit determines the polarity of the $\left(A/\sqrt{2}\right)g_{TX}(t)\sin(2\pi f_c t)$ term (the *quadrature* term).

To create the transmitted pulse represented by Eq. (1) or (3), we must provide the USRP with the complex baseband pulse

$$\tilde{x}(t) = Ag_{TX}(t)e^{j\theta}, \quad \theta = -3\pi/4, -\pi/4, \pi/4, 3\pi/4. \tag{4}$$

In this case the *symbol* is the complex number $Ae^{j\theta}$. The USRP will generate the transmitted signal of Eq. (1):

$$x(t) = \mathrm{Re}\left[\tilde{x}(t)e^{j2\pi f_c t}\right] = Ag_{TX}(t)\cos(2\pi f_c t + \theta). \tag{5}$$

A convenient way of visualizing the complex signal represented by Eq. (4) is to plot the possible symbol values $Ae^{j\theta}$ in the complex plane. A *signal constellation* for QPSK is shown in Figure 1.

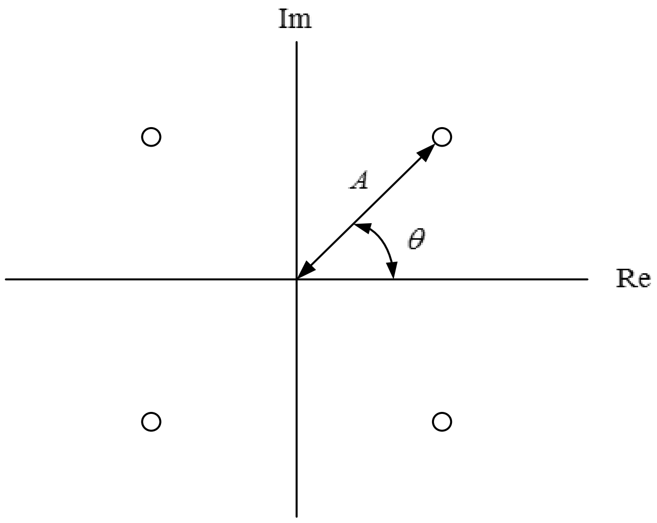

Figure 1. QPSK Signal Constellation

It is easy to see from the signal constellation that there are four possible transmitted signals (hence two bits per symbol), that each of the possible symbols has the same amplitude A, and that the four possible phase angles are $\theta = \pm\pi/4, \pm 3\pi/4$.

Let us now examine the relation between pairs of input bits and points in the signal constellation (i.e., symbol values). The mapping of bit pairs to points in the signal constellation is in fact arbitrary, as long as the transmitter and receiver use the same mapping. If we associate one bit with the polarity of the real part (in-phase term) and one bit with the polarity of the imaginary part (quadrature term) we obtain the mapping shown in Figure 2. As shown in the figure, the right-hand bit governs the polarity of the real part, while the left-hand bit governs the polarity of the imaginary part. It will be convenient later on to interpret the bit pairs as binary numbers. In doing this, we will interpret the left-hand bit as the *least* significant bit, and the right-hand bit as the *most* significant bit. This interpretation may seem contrary to intuition, but these bit sequences will be stored as arrays, and it is natural to write arrays with the index increasing to the right. The symbol mapping can also be shown in a table. Table 1 is the symbol mapping of Figure 2.

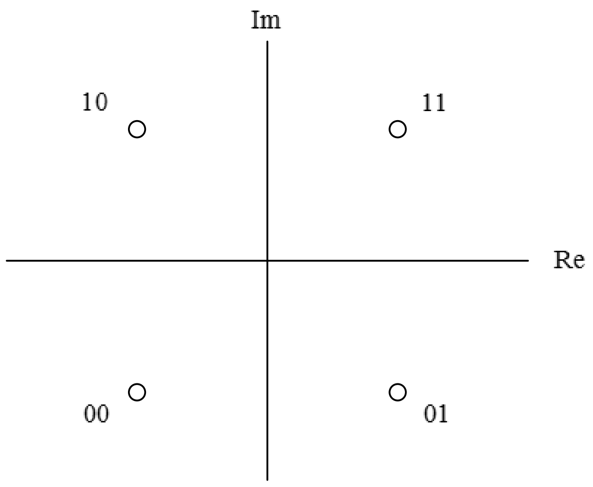

Figure 2. A Possible Symbol Mapping

Table 1. Symbol Mapping

Index Number	Bit Pattern	Symbol Value
0	00	$Ae^{-j3\pi/4} = -A/\sqrt{2} - jA/\sqrt{2}$
1	10	$Ae^{j3\pi/4} = -A/\sqrt{2} + jA/\sqrt{2}$
2	01	$Ae^{-j\pi/4} = A/\sqrt{2} - jA/\sqrt{2}$
3	11	$Ae^{j\pi/4} = A/\sqrt{2} + jA/\sqrt{2}$

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

$$r(t) = Dg_{TX}(t)\cos(2\pi f_c t + \theta + \phi), \tag{6}$$

where D is a constant (usually much smaller than the constant A in the transmitted signal), the angle θ carries the information, 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 QPSK signal. The receiver's *Fetch Rx Data* will provide a train of output pulses, each given by[8]

$$\tilde{r}(t) = \frac{D}{2} g_{TX}(t) e^{j(\theta+\phi)}. \tag{7}$$

The pulse train represented by Eq. (7) is passed through a matched filter having impulse response $g_{RX}(t)$. If we write $g(t) = g_{TX}(t) * g_{RX}(t)$, then the matched filter output can be written

$$\tilde{y}(t) = \frac{D}{2} g(t) e^{j(\theta+\phi)}. \tag{8}$$

The matched filter output is then sampled, at the rate of one sample per symbol. Ideally $g(0)=1$, so the sampler output is a train of complex numbers of the form

$$\tilde{y}[k] = \frac{D}{2} e^{j(\theta_k+\phi)}, \tag{9}$$

where θ_k is the phase value at the sample time $t = kT$ and D and ϕ are constants. In practice, the samples $\tilde{y}[k]$ will be corrupted by ISI and noise. As in the previous lab project, ISI is removed by an equalizer filter. As we have seen, the equalizer will also remove the phase error ϕ, leaving us with

$$\tilde{y}[k] = \frac{D}{2} e^{j\theta_k} + \text{ residual ISI} + \text{ noise}. \tag{10}$$

The remaining receiver steps are detection and symbol mapping. Signal detection is the process of examining each sample $\tilde{y}[k]$ and determining for that sample, which symbol is most likely to have been transmitted. It turns out that the algorithm for detection, called a *decision rule*, is surprisingly simple and has a nice geometric formulation. Figure 3 shows the signal constellation with the sample $\tilde{y}[k]$ also shown. Owing to residual ISI and noise, $\tilde{y}[k]$ does not correspond exactly to any of the possible symbol values.

[8] All of the baseband signals are actually represented in LabVIEW as discrete-time signals. Writing them as continuous-time signals makes the description given here much easier to read.

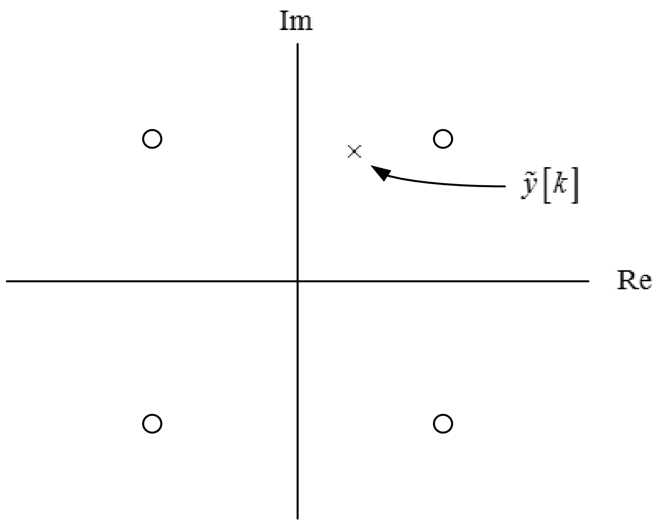

Figure 3. Signal Constellation and Received Sample

The decision rule for determining which symbol is most likely to have been transmitted given $\tilde{y}[k]$ is this: Calculate the distance between $\tilde{y}[k]$ and each possible symbol. The most likely symbol is the one at the smallest distance from $\tilde{y}[k]$. In Figure 3, the most likely symbol is the one in the first quadrant. Once the symbol has been determined, the symbol mapping of Table 1 can be used to convert the symbol to a bit sequence. For the example of Figure 3, the corresponding bit pattern is 11.

12.3 Pre-Lab

1. Create a function to perform the symbol mapping at the transmitter. (Detection and mapping at the receiver will be performed by *MT PSK Feedforward Equalizer*.) Table 2 gives the input and output specifications for your function.

Table 2. Specifications for Map Data

Inputs			Output		
Name	Type		Name	Type	
Data In	1-D array of 8-bit integer	Data bit stream	Symbols Out	1-D array of double complex	Symbol stream to be transmitted
Symbol Map	1-D array of double complex	Symbols indexed by bit-pairs represented as numbers			

Your function must conform to the symbol mapping of Table 1 with $A=1$. A template *MapDataTemplate.gvi* has been provided with the inputs and outputs already wired. The symbol map array that you will need as one of the inputs is available in the template *QPSKTx.gvi* described below.

Save your symbol mapping function in a file whose name includes the letters "MapData" and your initials (e.g. *MapData_BAB.gvi*).

2. Create a program to implement the QPSK transmitter. A template *QPSKTxTemplate.gvi* has been provided to get you started. This template includes the inputs and outputs, the symbol map array, functions for signal and power spectrum display, and the functions needed to interface with the USRP.

You will need to add

- the symbol mapping function from Step 1 above,
- *AddFrameHeader(Complex)*
- Upsampling
- Root-raised-cosine pulse shaping
- Amplitude scaling

Tip: Use your BPSK transmitter as a model. The QPSK transmitter differs only in the symbol mapping.

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

8. Create an equalizer function for QPSK. You can do this by making a copy of your BPSK equalizer function and then making three modifications as follows:

   - Change the "M-PSK" input to *MT Generate System Parameters* from 2 to 4.
   - Replace the symbol map with a copy of the QPSK symbol map from *QPSKTxTemplate.gvi*.
   - Replace the training sequence array constant with a new training sequence in which every bit is doubled. Figure 4 shows an easy way to do this with an *Interleave 1D Arrays function*.

   Save your equalizer in a file whose name includes the letters "QPSK_Equalizer" and your initials (e.g. *QPSK_Equalizer_BAB.gvi*).

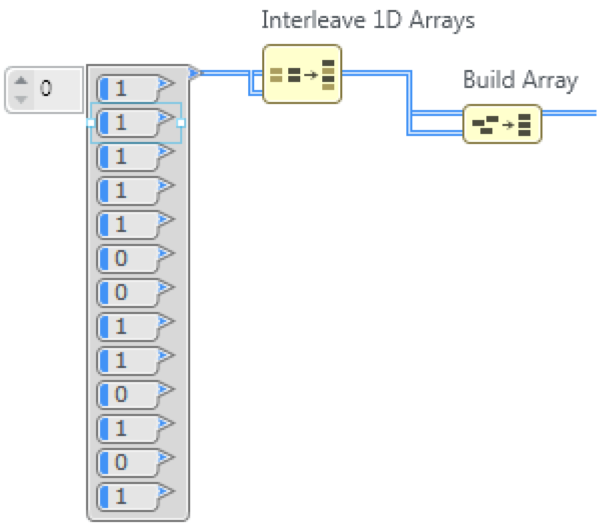

Figure 4. Modifying the Training Sequence

4. Create a program to implement the QPSK receiver. A template *QPSKRxTemplate.gvi* has been provided to get you started. The template includes the inputs and outputs, the functions needed to interface with the USRP, and several functions for configuring displays.

   Tip: Use your BPSK receiver as a model. The QPSK receiver is very similar.

You will need to add

- *Channel.gvi*.
- Use a Cluster Properties function after the *Channel.gvi* to give access to its "Y" component, an array of double complex sample values.
- Root-raised-cosine matched filter and the *Convolution* function to implement the filter.
  Note: There is no phase synchronizer, since this function will be performed by the equalizer. Also, do not take the real part of the received sample values.
- Using a Cluster Properties function, create a complex double cluster from the output of *Convolution*.
- *PulseAlign (Complex)*.
- Use a Cluster Properties function after the *PulseAlign(Complex)* to give access to the "Y" component. Add *Decimate (single shot)* to sample the aligned pulses.
- *FrameSync(Complex)*.
- Your equalizer from Step 3 above comes next. The "Output Signal" from *FrameSync(Complex)* is the equalizer input. Do not use the "Aligned Samples" *FrameSync(Complex)* output.
- Connect the "Output Bits" equalizer output to an *Array Subset function*. Set the "index" to 52 to remove the training sequence header and set the "length" to the number of message bits. The output of *Array Subset function* can be displayed as the receiver's "Output Bits" and can also be sent to *MT Calculate BER*.
- There are two *MT Format Eye Diagram* functions in the template. Connect the first one to the "Output Signal" from *PulseAlign(Complex)*. This will display the eye diagram of the in-phase component of the received signal before equalization. Connect the second eye diagram to "Output Complex Waveform" from the equalizer. This will display the eye diagram of the in-phase component of the received signal after equalization.

Also connect the "Output Complex Waveform" from the equalizer to the "waveform" input of *MT Format Constellation*.

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

Questions

1. The symbol rate of your transmitter and receiver is 10,000 symbols/second. What is the data rate in bits/second?
2. If the receiver is set for an IQ rate of 200 kHz, how many samples per symbol will there be at the receiver?
3. The *spectral efficiency* of a modulated signal is the data rate in bits/second divided by the bandwidth of the transmitted signal. Calculate the spectral efficiency of the BPSK signal using root-raised-cosine pulses that you generated in the BPSK lab project. Compare with the spectral efficiency of a QPSK signal, assuming that the bandwidth remains the same.

12.4 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

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

4. Using the power spectrum display on the transmitter front panel, measure the bandwidth of the complex baseband signal.

5. Ensure that the receiver is set up to use

   Carrier Frequency: 915.0 MHz

   IQ Rate: 200 kHz. Note: This sets the value of $1/T_z$. Note that T_z is the same parameter as dt .

   Gain: 0 dB

   Active Antenna: RX2

   Symbol rate: 10,000 symbols/s

   Message Length: 1000 bits

   Pulse shaping filter: Root Raised

   Use Channel: off

6. Run the transmitter, then run the receiver. Once the receiver has acquired its data, you may stop the transmitter. The receiver should show a BER of 0.0.

7. Compare the eye diagram before and after equalization. Run the receiver a dozen times or so until you get a feel for what changes from run to run and what does not. Occasionally the "before" eye diagram will show four sample values rather than two. Take screenshots of the "before" and "after" eye diagrams for this case.

8. Set Use Channel to "on," with the default channel model, and a propagation delay of $100\ \mu s$. Run the transmitter and then run the receiver several times. Compare the eye diagrams before and after equalization. Measure the eye opening before and after equalization. Try to get a worst-case "before" scenario for your measurement.

9. Set Use Channel to "off," and run the transmitter and receiver again. This time observe the signal constellation. Take a screenshot of the constellation.

10. Change the wiring to *MT Format Constellation* so that the input is taken from the *FrameSync(Complex)* "Output Signal" instead of from the equalizer. Run the transmitter and then run the receiver several times. What happens to the angle of the signal constellation? What happens to the radius of the signal constellation?

Questions

1. What is the measured bandwidth of the transmitted QPSK signal? (Note: Not the baseband signal bandwidth.) Compare with the bandwidth of the BPSK signal that you measured in an earlier lab project.

2. Report your eye opening measurement results from Step 6.

3. What is the cause of the signal constellation rotation that you observed in Step 8? What does the signal constellation look like when the "before" eye diagram shows four sample values rather than

two? Explain why the eye diagram appears as it does. Hint: Remember that the eye diagram plot displays only the in-phase component of the signal.

4. What would the signal constellation look like if the transmitter and receiver carrier frequencies were slightly different?

9. What would you expect to be the effect of rotation of the signal constellation on the BER? (Assume that the equalizer is not present.)

10. Observe the transition traces on the signal constellation graph. Notice that transitions between first and third quadrant signals pass through the origin of the graph, as do transitions between second and fourth quadrant signals. Describe what is happening to the amplitude of the received signal when these transitions occur.

12.5 Report

Prelab

Hand in documentation for the four functions you created: the symbol mapping, the transmitter, the equalizer, and the receiver. Also include documentation for any functions you may have 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 symbol mapping, transmitter, equalizer, and receiver programs. Resubmit documentation for any functions you modified during the lab.

Submit the plots asked for in Steps 5 and 7.

Answer the questions in the Lab Procedure section marked *Questions*.