Assignment: QPSK

Yangruo Ma, David Armstrong

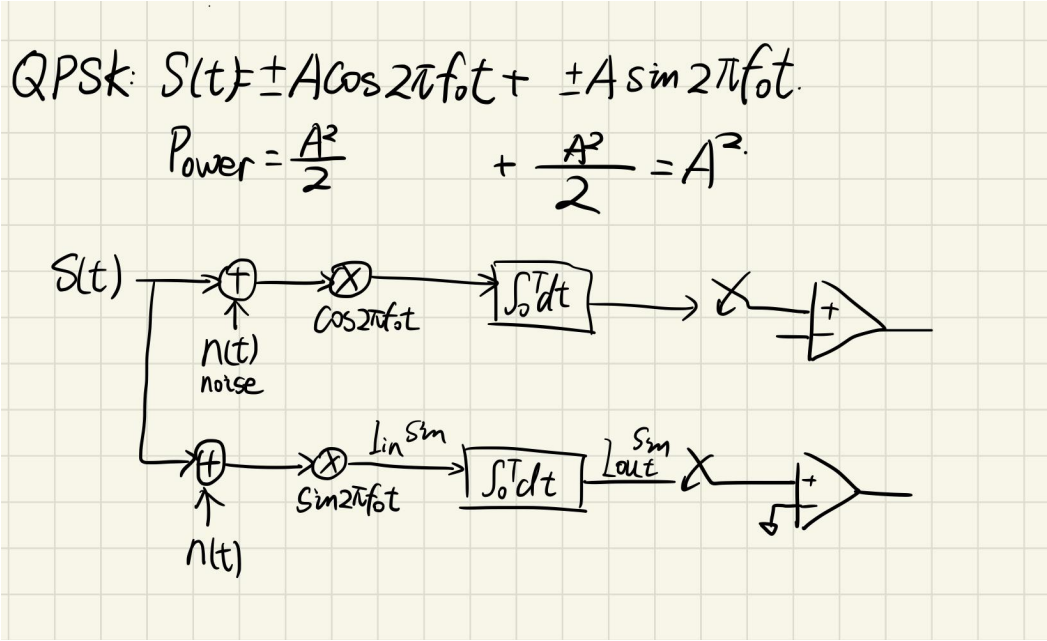
---

1. Abstract

In this lab, we simulate a Quadrature Phase Shift Keying (QPSK) system to analyze the advantages of QPSK over Binary Phase Shift Keying (BPSK). We also introduce noise into our theoretical system to see how the normalized signal to noise ratio (Eb/No) affects the accuracy of the QPSK process. We simulate the QPSK process for Eb/No values from -4 to 10 to create a histogram of fifteen different positive and negative integrator outputs and measure the error to compare against the theoretical output.

2. Introduction

QPSK is a way for a signal to carry twice as much information as BPSK by phase-shifting our signal in two different ways to have four different combinations of outputs instead of two. The diagram below shows the QPSK process and how combining two signals calls for twice the amount of power as one.



The Eular relationship is explained:

$$\sin \omega t = \frac{e^{j\omega t} - e^{-j\omega t}}{2j} \qquad \cos \omega t = \frac{e^{j\omega t} + e^{-j\omega t}}{2}$$

Then multiplen two sine wave:

$$\sin^2 \omega t = \frac{e^{j\omega t} - e^{-j\omega t}}{2j} \cdot \frac{e^{j\omega t} - e^{-j\omega t}}{2j} = \frac{e^{2j\omega t} - 2e^{0} + e^{-2j\omega t}}{-4}$$

$$= \frac{1}{2} - \frac{1}{2}\cos 2\omega t$$

From the above, multiplying two sine waves results in an output frequency of twice the input frequency (half the amplitude) superimposed on a DC offset of half the input amplitude.

Similarly, we multiply sin ωt by cos ωt to get:

$$\sin \omega t \cdot \cos \omega t = \frac{e^{2j\omega t} - e^{-2j\omega t}}{4j} = \frac{1}{2}\sin 2\omega t$$

This makes the output frequency (sin 2ωt) twice the input with no DC offset.

$$\sin\omega t \cdot \sin(\omega t+\phi) = \frac{e^{j\omega t}-e^{-j\omega t}}{2j} \cdot \frac{e^{j(\omega t+\phi)}-e^{-j(\omega t+\phi)}}{2j}$$

$$= \frac{e^{j(2\omega t+\phi)} - e^{j(\omega t-\omega t-\phi)} - e^{j(\omega t+\phi-\omega t)} + e^{-j(2\omega t+\phi)}}{-4}$$

$$= \frac{\cos(2\omega t+\phi)}{-2} - \frac{e^{j\phi}+e^{-j\phi}}{-4}$$

$$= \frac{\cos(2\omega t+\phi)}{-2} + \frac{\cos\phi}{2}$$

$$= \frac{\cos\phi}{2} - \frac{\cos(2\omega t+\phi)}{2}$$

$$\cos\omega t \cdot \sin(\omega t+\phi) = \frac{e^{j\omega t}+e^{-j\omega t}}{2} \cdot \frac{e^{j(\omega t+\phi)}-e^{-j(\omega t+\phi)}}{2j}$$

$$= \frac{e^{j(2\omega t+\phi)} - e^{j(-\phi)} + e^{j(\phi)} - e^{-j(2\omega t+\phi)}}{4j}$$

$$= \frac{\sin(2\omega t+\phi)}{2} + \frac{e^{j\phi}-e^{-j\phi}}{4j}$$

$$= \frac{\sin(2\omega t+\phi)}{2} + \frac{\sin\phi}{2}$$

We consider QPSK to be a form of phase-shift keying in which two bits are modulated at a time, selecting one of four possible carrier phase shifts (0, 90, 180, or 270 degrees). QPSK allows the signal to carry twice as much information as normal PSK using the same bandwidth. Two components are then generated: a cosine waveform with twice the received frequency, and a frequency-independent term whose magnitude is proportional to the cosine of the phase shift. With QPSK, the carrier undergoes four phase-changes (four symbols), so each symbol can represent 2 binary bits of data.

3. Design

We implement the QPSK system in NI LabVIEW by first making a sub-VI to run for one set of parameters. This sub-VI uses parameters such as signal amplitude, sampling information, Eb/No, iterations, to transform s(t) into an output centered around $\pm\ AT/2$. This is accomplished by multiplying the noisy s(t) signal by sine and cosine along two different paths, integrating each to Figure 1 below shows an overall block diagram of this sub-VI.

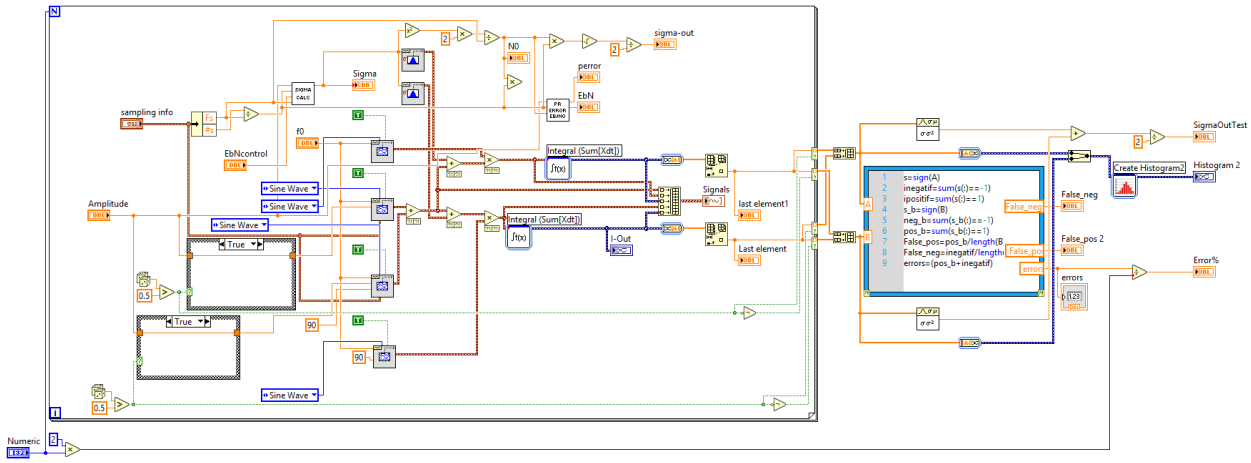


Figure 1: QPSK sub-VI expanding on initial BPSK rough draft. In brief, the VI generates a sine and cosine wave, combines them to form s(t), multiplies s(t) by sine along one path and cosine along another path, integrates each path, then plots the final point of the integration onto a histogram centered around $\pm\ AT/2$.

In more detail, for each iteration, the sign of the amplitude of each term of s(t) is randomly generated. S(t) is then formed according to f_0, the randomly "signed" amplitude, and the sampling information (kept at 1000 samples at 1kHz → 1 second sample). The standard deviation of the noise is calculated using a sub-VI "Sigma Calc," which is fed Amplitude, Time, and Desired-Eb/No to calculate the desired standard deviation of the noise. The noisy signal is then multiplied by a cosine and sine wave along different paths, integrated, and then visualized in the histogram below.

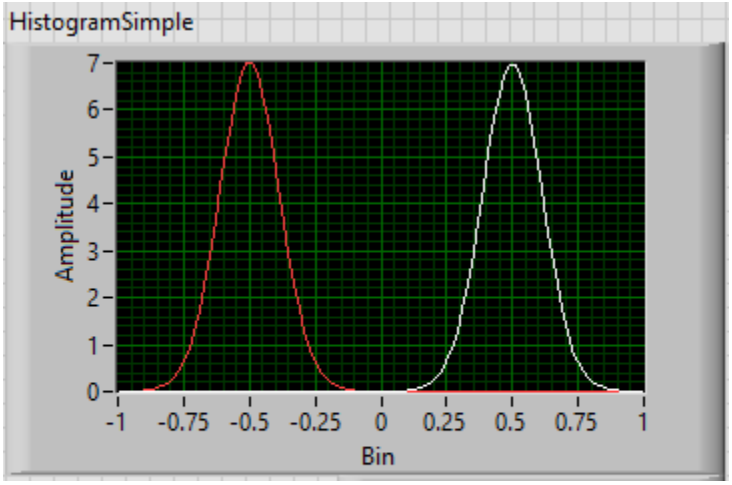


Figure 2: Histogram showing a QPSK system with very little error, as seen by the minimal overlap in the two sides. This is the output of the final Eb/No value, 10, with very small standard deviation in the noise used.

The integration averages an output of ±AT/2, which in our case stays at ±½. However, with wide levels of noise, this final value of the integration can be off by a factor of more than AT/2, causing false positives and negatives in the final reading. Our sub-VI counts how often these false readings occur (referenced from the amplitude boolean) and calculates the error rate using an imbedded MATLAB script. The deviation in the integral caused by noise can be seen in figure 3 below.

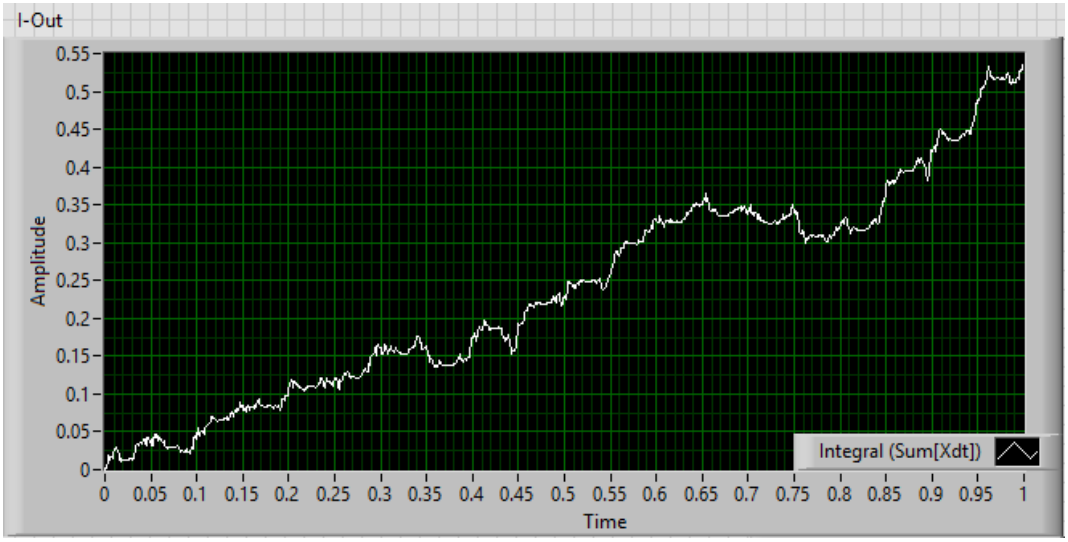


Figure 3: A closer look at the integral converging on AT/2 for a single iteration of QPSK. The random noise causes some of the inputs to be negative but on average, the thousand points will converge to AT/2.

Our design allows for the user to input a value of Eb/No, and then the standard deviation of the noise is changed appropriately. We chose this implementation over changing the amplitude because this allows each histogram to be consistently centered around the same AT/2. Figure 4 below is a visual summary of the QPSK process. In this visual, we can see the progression from a pure wave, to a noisy wave multiplied by the two different local oscillators, to the integration of these signals approaching AT/2.

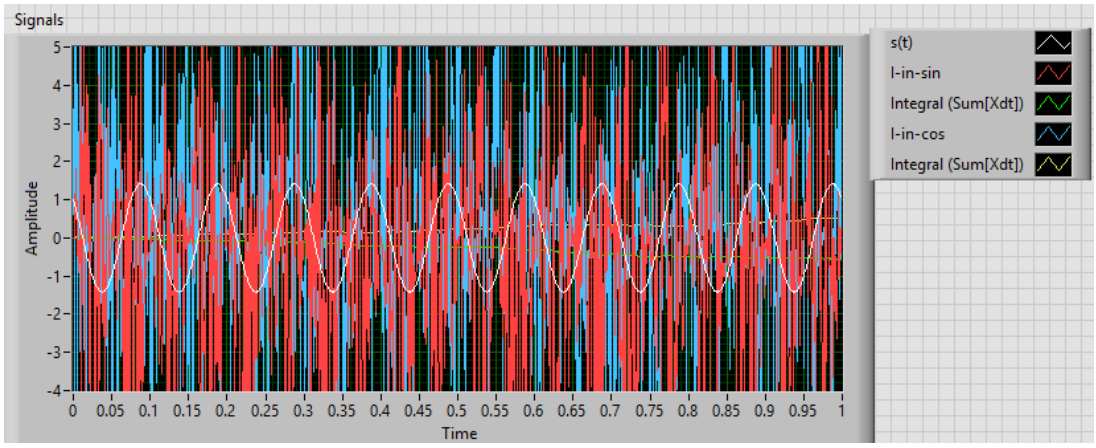


Figure 4: As described in the legend, this graph shows our initial s(t) in white, the combination of s(t), noise, and our signal multiplied by the local oscillator (sine or cosine) in red or blue, and the integral of each path approaching ±AT/2.

4. Operation

In order to visualize the accuracy of QPSK at various values of Eb/No and create our desired plots, we created an over-arching VI to run the QPSK routine repeatedly. This main VI iterates through Eb/No values -4 through 10 and specifies different iteration amounts for each Eb/No value. Each individual histogram is combined into a single graph to see how the higher standard deviation values cause the histogram to cross the threshold where the output of the integral is 0 and causes errors. The block diagram below shows how we automated the operation of our QPSK sub-VI.

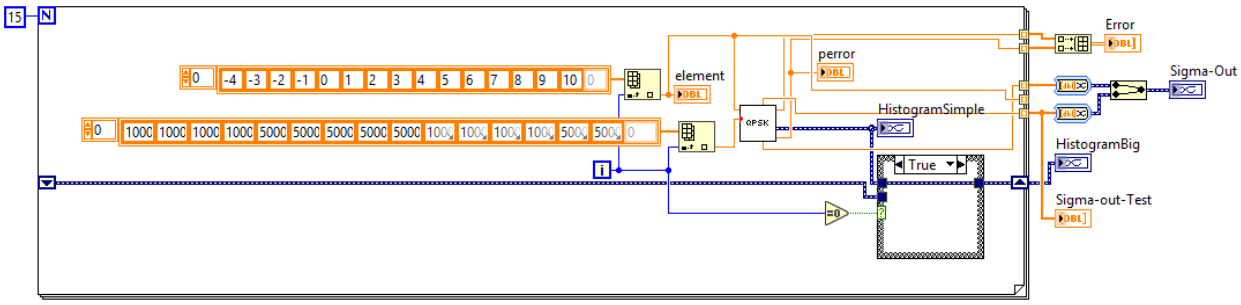


Figure 5: The QPSK demonstration VI, testing values of Eb/No by changing sigma each time while running each QPSK sub-VI an increasing number of times to reach an accurate testing standard deviation and error rate in the output histograms.

This operation results in a summary histogram displaying the narrowing of the simulation curves as No decreases. Figure 6 aptly displays the objective concept of the error increasing along with the standard deviation.

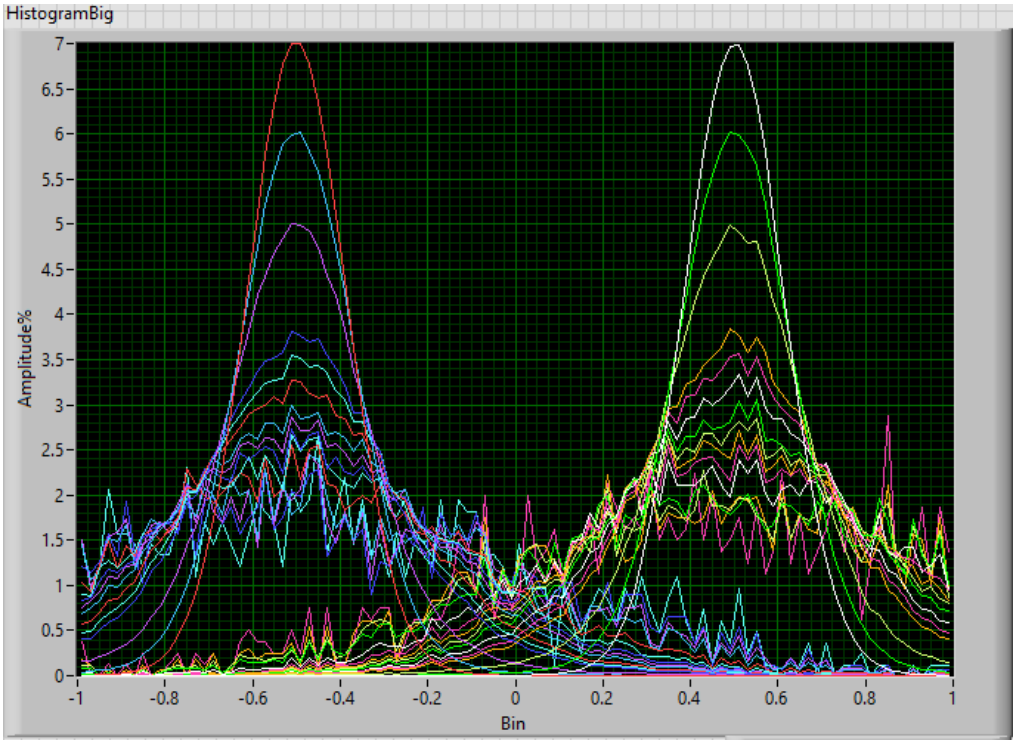


Figure 6: Combined histogram of many QPSK iterations. The red and white curves at the top represent the low-error, small-deviation outputs of Eb/No = 10, while the cyan and pink curves at the bottom represent the high-error, large-deviation outputs of Eb/No = -4. Each bin is a different amplitude output of the final integration.

If our implementation increased Eb/No by increasing the initial amplitude "A", this histogram plot would instead be many histograms side by side at different AT/2 values. The decreasing "smoothness" is caused by the values being more spread out as well as fewer iterations being run at small values of Eb/No, resulting in fewer average samples per bin. These wider curves are seen at the start of figure 7 below with higher error rates than their less noisy peers.

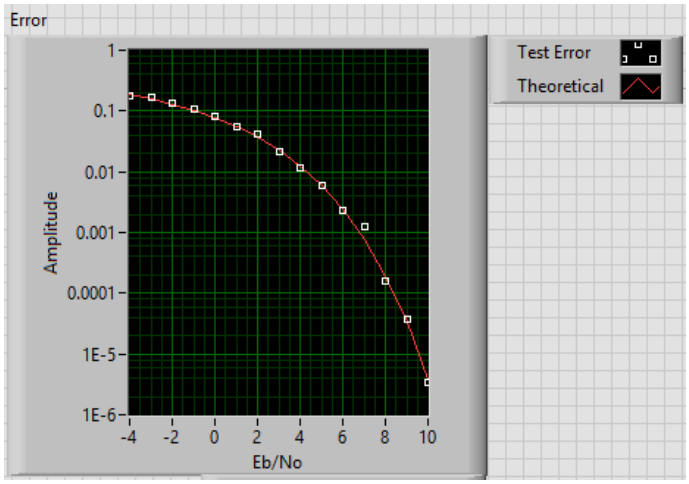


Figure 7: Theoretical vs test data error for Eb/No values from -4 to 10. The low values of Eb/No imply high noise levels, causing many false negatives and false positives. The minimal error at high Eb/No values implies that the QPSK system works almost-perfectly with the almost-absence of noise.

The standard deviation of the output is descending as expected. Since Eb/No is increased by decreasing "No" which is square-rootly proportional to standard deviation, our standard deviation decreases with Eb/No increasing.

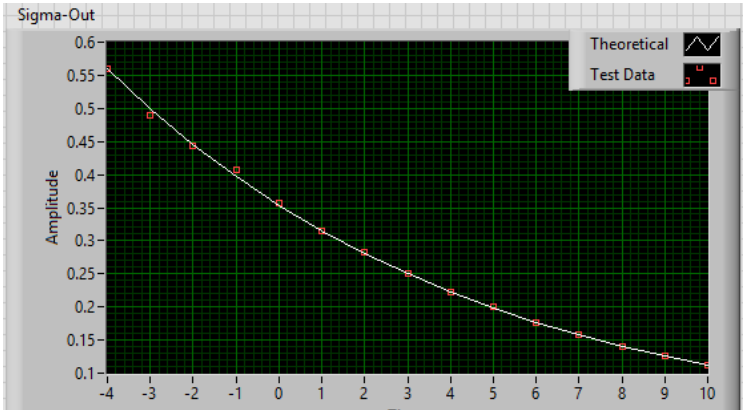
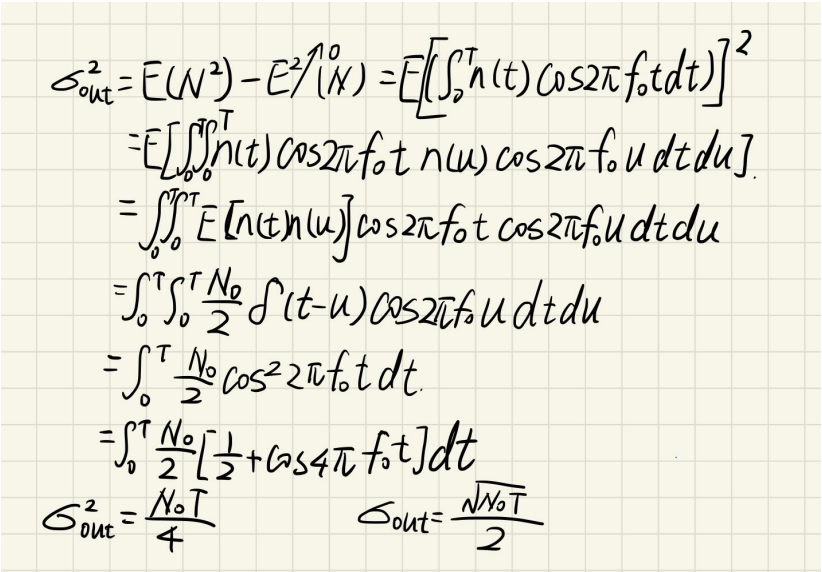


Figure 8: Theoretical vs test data variance in the final integration values for each Eb/No value from -4 to 10. A large Eb/No value was achieved through a small amount of deviation in the system noise which caused the final histograms to be skinnier, thus creating this negative-sloping curve. If our implementation increased Eb/No by increasing the initial amplitude "A", this variance plot would instead be a straight line.

With enough iterations, the measured standard deviation of our outputs matched the theoretical value, as derived below. In order for our error and standard deviation measurements to match the theoretical values, we had to adjust the number of iterations that were run for each Eb/No value. Since the higher values of Eb/No required so many iterations for the measured error to match the theoretical line, those higher Eb/No values match the standard deviation plot in figure 8 perfectly. On the other hand, the lower

Eb/No values start to deviate from the theoretical standard deviation line because they run for fewer iterations (because fewer were needed to match the theoretical error line).



5. Conclusion

QPSK actually transmits two bits per symbol. QPSK symbols do not represent 0 or 1, but 00, 01, 10, or 11. Compared to modulation schemes that transmit one bit per symbol, QPSK offers advantages in terms of bandwidth efficiency. For example, consider an analog baseband signal in a BPSK system, but BPSK uses two possible phase shifts instead of four, so only one bit can be transmitted per symbol. The baseband signal has a certain frequency, and in each symbol period, one bit can be transmitted. A QPSK system can use the same frequency baseband signal, but it transmits two bits per symbol period. So it's (ideally) twice as bandwidth-efficient.

Our design is quality in the fact that our outputs function as intended and the operation VI is easy to use and understand. However, improvements could be made in terms of coding practices, and accuracy. In terms of accuracy, some of our error and standard deviation points did not land exactly on the theoretical curve, which could be fixed with a simple increase in the number of iterations. In terms of coding practices, many aspects could have been cleaner. The iteration amount had to be manually inputted from an array because we could not think of a satisfactory function to map Eb/No or the index to an appropriate iteration amount. Our MATLAB code is slightly difficult to understand and possibly could have been accomplished using LabVIEW functions instead to create a more visually comprehensible solution. But overall, we are satisfied with the result and complexity of our work. This lab taught us the details of QPSK implementation, the real-life consequences of changes in noise and signal power, and improved our LabVIEW skillset.