Simulation and Optimization of QPSK System Performance over AWGN Channels

Biqi Liu, Siying Chen

1 Abstract

This report evaluates the performance of a QPSK communication system under an AWGN channel through simulation. The BER performance is analyzed and compared with theoretical values. Results show that at high E_b/N_0, the simulated BER matches theory, while at low E_b/N_0, deviations occur due to synchronization and phase estimation errors. Timing estimation is more noise-sensitive. Differential QPSK (D-QPSK) is also tested, showing robustness to phase errors but worse BER. Improvements such as longer training sequences and better synchronization algorithms are suggested.

2 Background and Problem Formulation

QPSK (Quadrature Phase Shift Keying) is widely used due to its balanced trade-off between spectral efficiency and robustness against noise. Below is the simulation of the QPSK system, including the transmitter, AWGN channel, and receiver.

2.1 Transmitter

The transmitter starts by appending guard bits, training bits, and data bits to the transmission. Guard bits are added to ensure that the transients at the beginning and end of the received sequence do not interfere with the sampled signals. Training bits are predefined and known to the receiver, allowing it to synchronize and adjust for phase shifts.
The transmission process begins by mapping every pair of bits to QPSK symbols using Gray coding. Following this, pulse shaping is applied using a rectangular filter, resulting in a baseband signal.
In a real system, the baseband signal would be upconverted to a higher carrier frequency, but in this simulation, we skip that step, operating directly in the baseband.

2.2 Channel

The channel is modeled using an AWGN channel, where Gaussian white noise is added to the transmitted signal to simulate noise interference.

2.3 Receiver

In the receiver, the signal undergoes bandpass and lowpass filtering to remove noise and downconvert to baseband, although upconversion is skipped in this simulation. The filtered signal is passed through a matched filter, designed to match the pulse shape used in transmission, highlighting the peak of the signal and aiding synchronization. After synchronization, the receiver samples the QPSK symbols at the correct rate, compensating for phase offsets by comparing the received training symbols with the known ones. The final step is decision making, where the received symbols are mapped back to their corresponding 2-bit sequences.
Since Matlab does not support continuous-time processing, we simulate continuous-time signals by oversampling. In this simulation, we use 8 samples per symbol to effectively model the continuous-time behavior of the system.

3 Methodology

3.1 Simulated BER and Derivation

Since Gray coding is used, each pair of bits corresponds to a QPSK symbol. The odd-indexed bit maps to the real part of the symbol, while the even-indexed bit maps to the imaginary part. Consequently, when analyzing bit error probability (BER), we can consider it as the probability of incorrectly detecting either the real or imaginary component of the QPSK symbol. By leveraging the Gray-coded constellation diagram, we can derive the BER expression accordingly.

$$\begin{aligned} P_b &= P(\text{wrongly detect in one dimension}) \\ &= Q(\frac{\frac{d}{2}}{\sqrt{\frac{N_0}{2}}}) = Q(\frac{d}{\sqrt{2N_0}}) = Q\left(\sqrt{\frac{2E_b}{N_0}}\right) \end{aligned} \tag{1}$$

The simulation results agree with the theoretical results, as shown in figure 5

3.2 Performance of Phase and Timing Estimation

When E_b/N_0 is very large, both timing and phase estimation are accurate, with errors close to zero. Timing estimation is more sensitive to noise because noise distorts the correlation peak, making it harder to find the correct sampling point. Phase estimation is less sensitive as it averages phase over many symbols. Timing estimation can be improved by using a longer training sequence or better synchronization algorithms, while phase estimation can be improved with a Decision-Directed PLL.

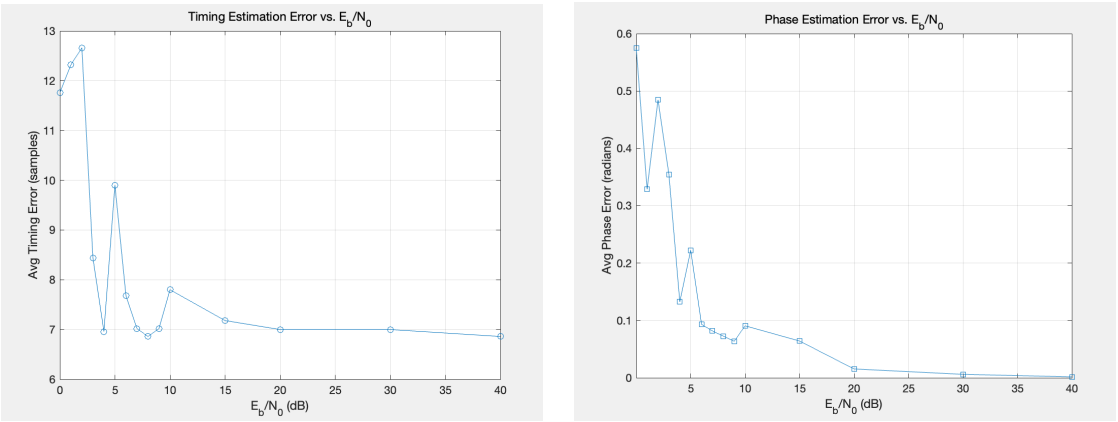

Figure 1: Average Error of timing and phase estimation

3.3 Received Signal Constellation

The transmitted signal is intended to lie precisely at the corners of the square constellation. However, when passing through the AWGN channel, the signal fluctuates around its original position, with its mean value remaining at the transmitted symbol's location. As the E_b/N_0 increases, the constellation points become more concentrated around their original positions.

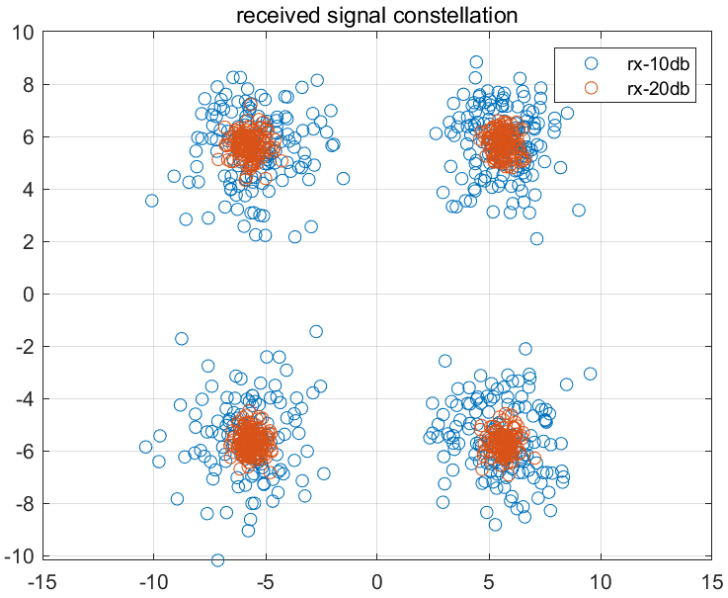

Figure 2: Received Signal Constellation for different SNR

3.4 Power Sprectrum Density

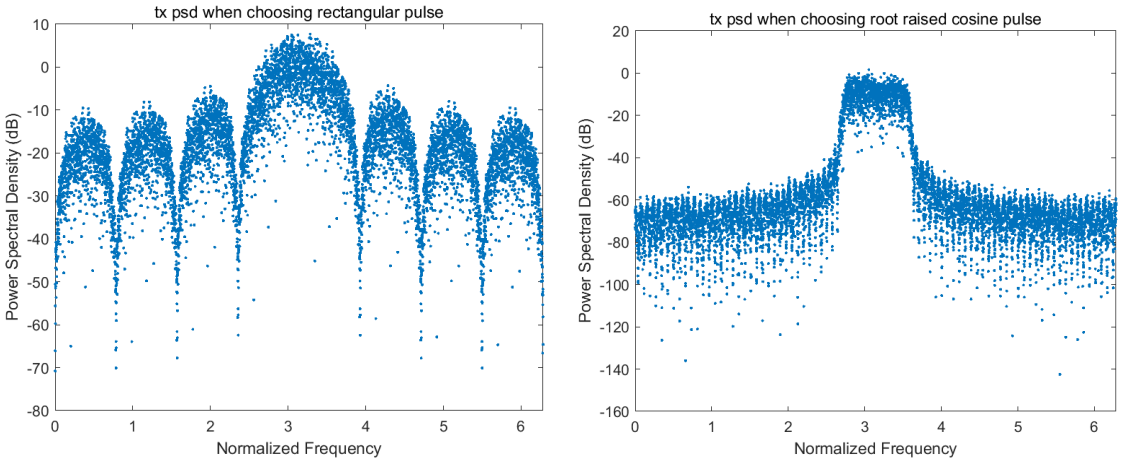

Figure 3: transmitted signal psd using different pulse

rectangular window
The rectangular window is simple, but it has high sidelobes in the frequency spectrum, which can lead to significant Inter-Symbol Interference (ISI), especially in scenarios with multipath propagation and limited bandwidth. However, it remains an effective choice in systems where simplicity, unconstrained bandwidth, and minimal ISI requirements are prioritized.
raised cosine window
The raised cosine window has almost no sidelobes in its frequency spectrum and offers high bandwidth efficiency. It is particularly effective at mitigating ISI. In modern communication systems, the raised cosine window is more commonly used due to its superior ISI control and bandwidth efficiency, especially in cases that require spectrum optimization or high bandwidth utilization.

3.5 DPSK

Differential Phase Shift Keying (DPSK)
DPSK does not require explicit synchronization because it transmits information by encoding data in the phase difference between consecutive symbols. In DPSK, if synchronization is reliable, the training sequence can be omitted since there is no need for separate phase or timing estimation.
Quadrature Phase Shift Keying (QPSK)
In contrast, QPSK requires a known training sequence to perform phase estimation, adding complexity to the receiver design.
When comparing Bit Error Rates (BER), DPSK typically exhibits a higher BER than QPSK. This is because DPSK's detection process involves two consecutive symbols, which compounds the noise variance, whereas QPSK detection is based on a single symbol, resulting in lower noise accumulation.

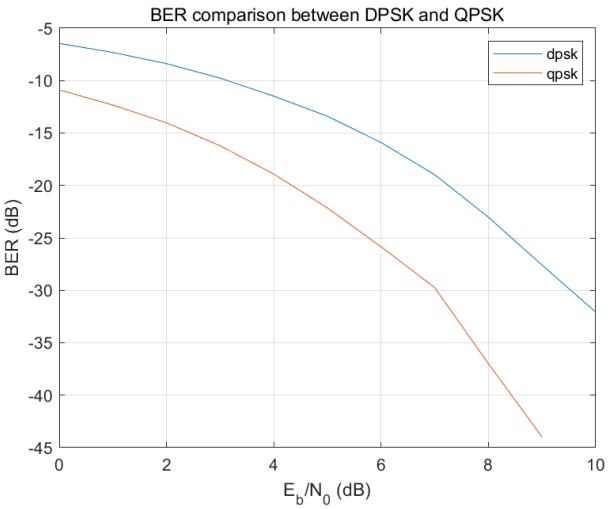

Figure 4: BER of DPSK and QPSK

4 Results

4.1 BER Performance Analysis of QPSK System

The simulated and theoretical BER results for the QPSK system over an AWGN channel are summarized in the table and figure below. The BER decreases as E_b/N_0 increases, aligning with the expected behavior of digital modulation schemes in noise-limited environments.

E_b/N_0 **(dB)**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Theoretical BER	0.0786	0.0563	0.0375	0.0229	0.0125	0.00595	0.00239	0.000977	0.000378
Simulated BER	0.0809	0.0583	0.0388	0.0235	0.0127	0.0063	0.0020	0.0010	0.0002

Table 1: Theoretical and Simulated BER Results for QPSK

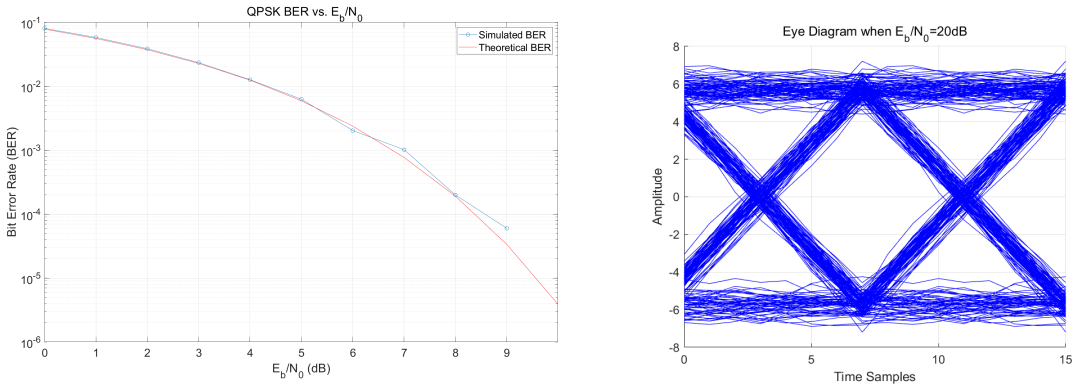

Figure 5: transmitted signal psd using different pulse

4.2 Observations and Explanations

Both simulated and theoretical BER curves follow the same exponential decay trend. At $E_b/N_0 = 0\,\text{dB}$, the simulated BER is higher than the theoretical value, while at $E_b/N_0 = 10\,\text{dB}$, the simulated BER reaches zero, closely matching the theoretical limit. This behavior aligns with QPSK's theoretical BER formula:

$$P_b = Q\left(\sqrt{2\frac{E_b}{N_0}}\right)$$

where the Q-function governs the error probability under AWGN.

When $E_b/N_0 < 5\,\text{dB}$, the simulated BER slightly exceeds the theoretical values. This may be due to:

- **Non-Ideal Synchronization:** Timing offsets during symbol sampling introduce intersymbol interference, degrading performance.
- **Phase Estimation Errors:** Residual phase offsets after correction rotate the signal constellation, increasing decision errors.
- **Finite Simulation Length:** Limited transmitted bits lead to statistical fluctuations in error counting.

When $E_b/N_0 \geq 8\,\text{dB}$, the simulated BER approaches zero, matching theoretical predictions. This indicates that synchronization and phase estimation algorithms perform adequately in high SNR regimes.

The deviation between simulated and theoretical BER primarily arises from practical non-idealities in the simulation process. The theoretical model assumes perfect synchronization, phase alignment, and noise variance estimation, whereas in simulations, timing synchronization and phase estimation are affected by noise, especially at low SNR, leading to sampling offsets and constellation rotation. Additionally, the use of a finite training sequence limits estimation accuracy. Discrete-time modeling further introduces quantization errors, pulse shaping effects, and potential mismatches in noise power normalization, particularly when mapping continuous-time noise variance N_0 to the sampled system. These factors collectively contribute to the discrepancies observed between the simulated and theoretical BER curves.

4.3 Potential Improvements

Enhance Synchronization Accuracy:

- Replace the cross-correlation-based synchronization with interpolated correlation-based timing estimation to reduce timing jitter, or consider maximum likelihood (ML) synchronization using the training sequence for improved robustness in low SNR conditions.
- Increase the training sequence length to improve synchronization accuracy and correlation peak detection.

Refine Phase Estimation:

- Implement a decision-directed phase-locked loop (DD-PLL) to adaptively track phase variations during data transmission, which is particularly effective if the channel exhibits slow phase drift or residual phase noise.

Increase Simulation Robustness:

- Increase the number of transmitted bits to reduce statistical uncertainty in BER estimation, especially at high SNR where error events are rare.
- Introduce pulse shaping (e.g., root-raised cosine filter) to reduce spectral sidelobes and improve robustness in band-limited or dispersive channels.

5 Conclusions

The simulation shows that the BER performance of the QPSK system closely follows the theoretical curve at high E_b/N_0 , but deviates at lowE_b/N_0 . This deviation is mainly due to non-ideal timing and phase estimation, which introduce sampling errors and phase rotation. Timing estimation is more sensitive to noise than phase estimation because noise distorts the correlation peak used for synchronization. Phase estimation is more robust as it averages over multiple symbols, though it can still be affected by noise at low E_b/N_0 .The signal constellation diagrams illustrate that noise causes symbol spreading, while phase estimation errors lead to constellation rotation. The power spectral density analysis shows that a root-raised cosine pulse reduces spectral sidelobes compared to a rectangular pulse. Differential QPSK (D-QPSK) is implemented as an alternative that does not require phase estimation. It is robust to phase errors but shows worse BER performance at low E_b/N_0 , primarily due to error propagation during differential decoding.To improve system performance, a longer training sequence can enhance synchronization accuracy, interpolation-based timing estimation can reduce sampling errors, and a decision-directed phase-locked loop (PLL) can adaptively track phase variations during data transmission.

Appendix

qpsk.m

```
function d = qpsk(b)
    l = length(b)/2;
    d = complex(zeros(1,l));
    for i = 1:l
        d_I = (b(2*i-1)==0)*2 - 1;
        d_Q = (b(2*i)==0)*2 - 1;
        d(i) = (d_I + 1i*d_Q) / sqrt(2);
    end
end
```

sync.m

```
function t_samp = sync(mf, b_train, Q, t_start, t_end)
    d_train = qpsk(b_train);
    len_train = length(d_train);
    max_cor = 0;
    max_pos = 0;
    for i = t_start:t_end
        matched_out = complex(zeros(1,len_train));
        for j = 1:len_train
            matched_out(j) = mf(i+(j-1)*Q);
        end
        xcorr_result = xcorr(d_train, matched_out);
        cor = max(abs(xcorr_result(len_train)));
        if cor > max_cor
            max_cor = cor;
            max_pos = i;
        end
    end
    t_samp = max_pos;
end
```

phase_estimation.m

```
function phihat = phase_estimation(r, b_train)
    d_train = qpsk(b_train);
    r_train = r(1:length(d_train));
    op_ph = 0;
    op_norm = norm(r_train - d_train);
    ph_range = 0:0.1:2*pi;
    for phase = ph_range
        r_rotate = r_train.*exp(1i*phase);
        diff_norm = norm(r_rotate - d_train);
        if diff_norm < op_norm
            op_norm = diff_norm;
            op_ph = phase;
```

```
        end
    end
    phihat = 0-op_ph;
end
```

detect.m

```
function bhat = detect(r)
    b_len = length(r)*2;
    result = zeros(1,b_len);
    for i = 1:length(r)
        result(2*i-1) = 0 + (real(r(i))<0);
        result(2*i) = 0 + (imag(r(i))<0);
    end
    bhat = result;
end
```