Cover-Page/Mark-Setting for Project Assignment
ANALYSIS AND SIMULATION OF A QPSK SYSTEM
EQ2310 Digital Communications

Filled-in by the student(s). State name and 'personnummer'.

Student 1: Giel Ooghe

Student 2: Jiangfan Feng

Check *one* of these alternatives:

Student 1 has worked alone, there is no Student 2	☐
Student 1 has written this report, but has worked together with Student 2	☐
Student 1 and Student 2 have worked and written this report together	☒

(*Fields below filled-in by the teacher.*)

	PASSED		FAILED	
length of report (number of pages)		≤ 6 pages ☐	> 6 ☐	
number of authors		≤ 2 ☐	> 2 ☐	
author name(s) and personal id number(s) stated		yes ☐	no ☐	
the text is easy to follow and the language correct	yes ☐	mostly ☐	sometimes ☐	no ☐
results and conclusions are clearly stated	yes ☐	most of them ☐	only a few ☐	no ☐
results are correct	yes ☐	most of them ☐	only a few ☐	no ☐
the extent of the work is satisfactory	yes ☐	almost ☐	no ☐	
the results can be reproduced	yes ☐	most of them ☐	no ☐	
the mathematics/theory is correct	yes ☐	mostly ☐	no ☐	hard to tell ☐

	P	F	
first check	☐	☐	Sign:
second check (when **F** above)	☐	☐	Sign:

PASSED

Signature:

EQ2310 Project Report

Jiangfan Feng, Giel Ooghe

February 9, 2023

1 Abstract

For this project, four Matlab functions were written in order to simulate a QPSK system. Using this a couple problems can be explored relating to the Bit Error Performance, the signal constellations, the Power Density Spectrum and the eye diagram. Furthermore a look is given at optimizing the length of the training sequence and its effects on synchronization and phase estimation.

2 Background and Problem Formulation

The first problem that was explored was an investigation of the bit error performance with perfect synchronization and phase estimation. Then a comparison is made with a theoretical estimate performance. This is necessary to see if the simulation performs as it's supposed to.

Secondly, the signal constellations in the receiver are compared for different values of Signal to Noise Ratios (SNR). The implications of phase estimate errors are also explained.

Then a plot is given of the Power Density Spectrums of the two given pulses (a block pulse and a root raised cosine pulse) and the effects of changing the pulses are explained.

Afterwards, a comparison of the eye diagrams is made for two different channels, the AWGN channel and the two-path ISI channel. This comparison looks at the effects on Bit Error Rate (BER), phase estimate accuracy and synchronization accuracy.

Lastly, a look is given at the effects of changing the length of the training sequence on the phase estimation and on the synchronization. What is a reasonable length and what would the corresponding overhead be.

3 Methodology

Problem statement 1:
For the BER, which represents the amount of bit errors in the transmitted message, it can be directly derived from the countered errors divided by the size of each data sequence and the number of simulated data sequence blocks.

Problem statement 2:
For signal constellation, the point is to change the level of SNR and observe the spread of the complex QPSK symbols.

Problem statement 3:
For power density spectrum, using matlab function "periodogram" of the transmitted signal to generate and compare the periodogram power spectral density.

Problem statement 4:
For this problem, the eye diagram has to be made and the BER, phase estimate accuracy and the synchronization accuracy have to be compared. To do so, the "eyediagram" command in matlab is used with 2 samples for each channel. For the other comparisons, the plots can be made using the two BER's, the two phihat's and the two

t_samp's

Problem statement 5:
Looping over the simulation with different training bits lengths gives different results for the BER, phase estimation and synchronization, combining the plot results for the different bit lengths gives a good estimate for the optimal training bit length.

4 Results

4.1 BER

Firstly, the exact expression for the corresponding BER is:

$$BER = Q\left(\sqrt{\frac{2E_b}{N_0}}\right) \tag{1}$$

It indicates that the data bits at the receiving end have a ratio of the difference to the original data. This can be compared by looking at this theoretical result of the BER and the result estimated through simulation. For this simulation an SNR was chosen between 0dB to 10dB (this is also the range in question 3). As is seen in figure 1, the simulated result and the theoretical result match pretty well, which means the estimated results agree with the simulation results, and a larger SNR could reduce the ratio error.

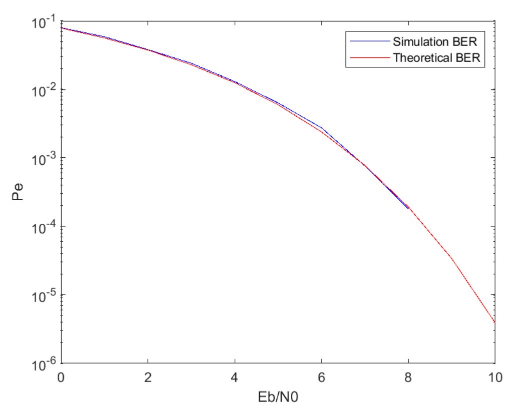


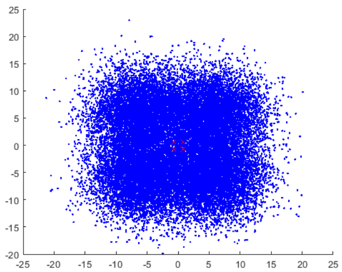
Figure 1: BER

4.2 Signal Constellation

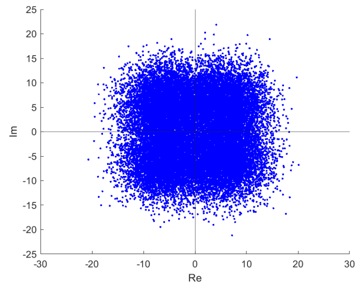
Studying the signal constellation in the receiver for various values of Eb/N0, the result is shown in figure 2. The selection of SNR value increases with the number of figures, and they range from 0dB to 10dB with 1dB difference each. The trend could be seen clearly from the figures that, when the larger SNR were selected, the more concentrated the points on the constellation would be. They tend to be distributed around the four corners of a square which is centered at the origin. Under 1dB, this trend could be hardly seen, which means that the noise affects a lot and the distribution is pretty random, which follows the finding of the BER comparison.

4.3 PSD

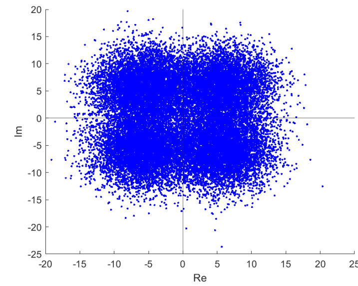
With the Matlab function, we could plot the power density spectrum of the transmitted signal with rectangular pulse and root raised cosine pulse, the two results are shown in figure 4. The main differences are that the root raised cosine pulse has an overall much lower power over frequency while the ones pulse only reaches low points at certain normalized frequency points. Those points are then the regular average depth of the root raised cosine function.



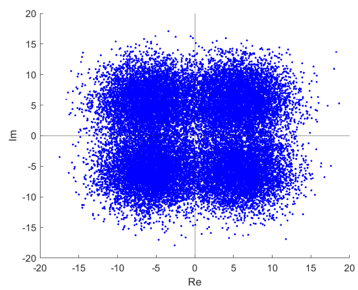
(a) SNR = 0dB



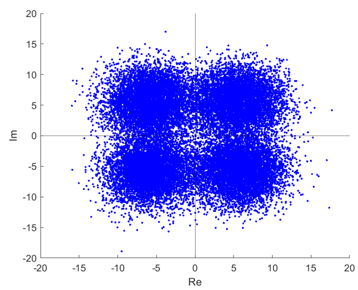
(b) SNR = 1dB



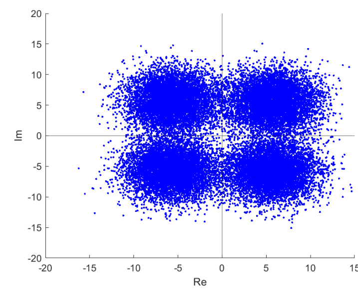
(c) SNR = 2dB



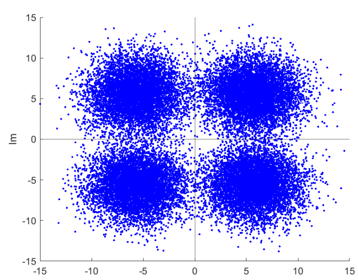
(d) SNR = 3dB



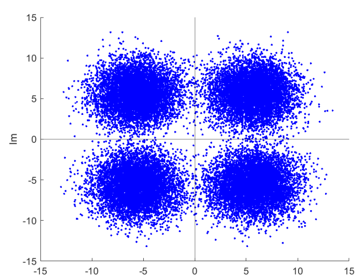
(e) SNR = 4dB



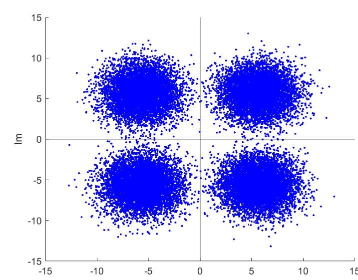
(f) SNR = 5dB



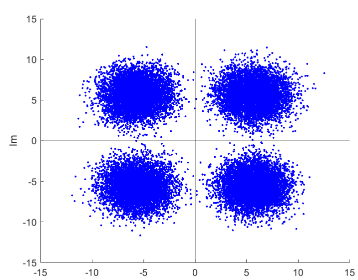
(g) SNR = 6dB



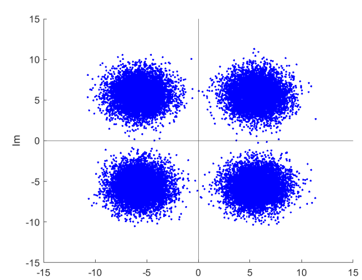
(h) SNR = 7dB



(i) SNR = 8dB

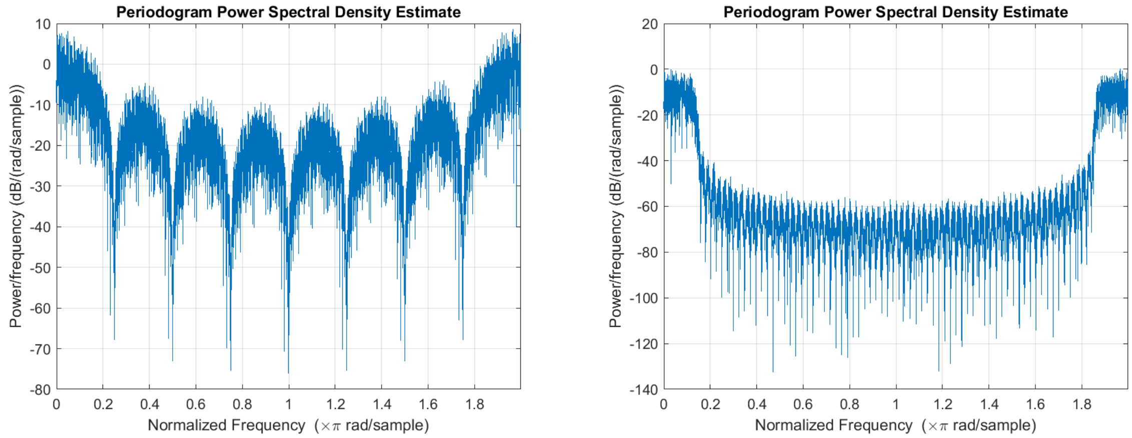


(j) SNR = 9dB



(k) SNR = 10dB

Figure 2: Constellations over 50 iterations with different SNR's



(a) Rectangular pulse

(b) Root raised cosine pulse

Figure 3: PSD's of Tx for different pulses

4.4 Eye Diagram

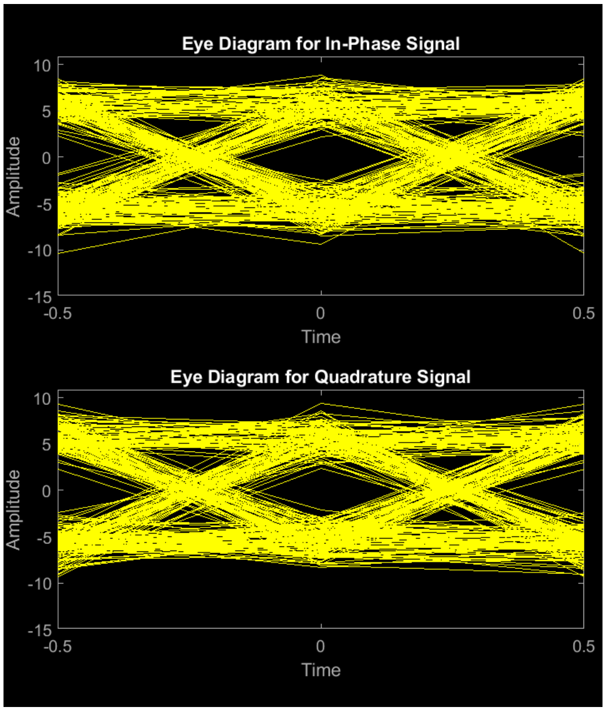
As is seen in Figure 4, the eye diagram of the AWGN channel is way better than the eye diagram of the two-path ISI channel. The opening is larger and the lines around it are way more distinct from each other. Whereas on the two-path ISI channel the opening is almost invisible and all lines criss-cross each other.
Figure 5a shows the effect on the Bit Error Rate. It is clear that the two-path ISI channel performs much worse than the AWGN channel, especially with high SNR values.
Figure 5b on the other hand shows the accuracy of the phase estimation. The two-path ISI channel seems to deviate a bit more than the AWGN channel, but overall the difference seems minimal. Figure 5c shows the difference in t_samp. These are completely the same throughout all values of SNR.

4.5 Length of training bits

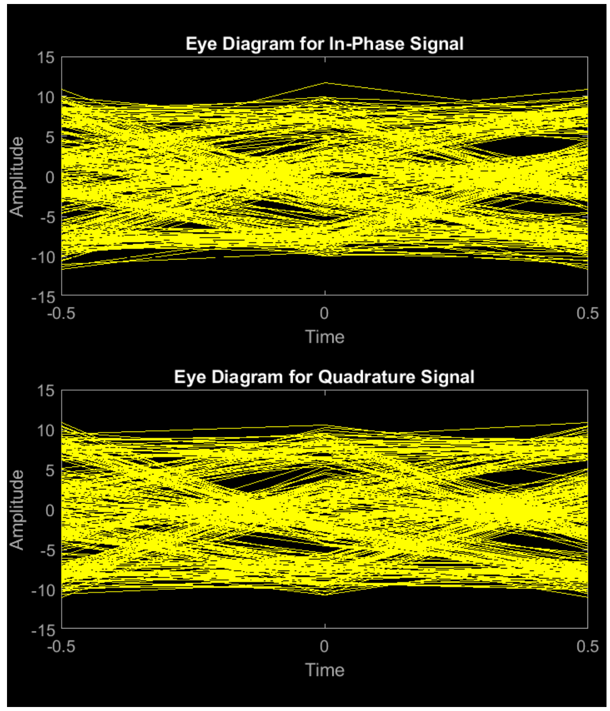
In figure 6a, it is possible to see the BER for different training bit lengths. Since there was a lot of data, a legend would have been very hard to read, but overall the highest lines are almost always smaller training bit lengths. The highest one is nr_training_bits = 0 bits and results in $P_e = 0.5$ and around nr_training_bits = 10 most of the graphs follow the theoretical graph. Then in figures 6c and 6b, the phase estimate and synchronization accuracy are plotted in function of nr_training_bits. Even here using a nr_training_bits = 10, seems to be a reasonable choice since most erroneous peaks happen with lower values. This results in an overhead of $10/1010 = 0.0099$.

5 Conclusion

During this exercise, some important aspects of baseband and passband simulations of a communication system were simulated. Based on this QPSK system, answers were investigated into the following five related problem topics: BER, Signal constellation, PSD, eye diagram and the optimal length of training bit sequence. Through this process, a practical understanding and a further appreciation of some of the descriptions in this course was gained. For example, in the BER part, one of the major lessons learned during this project was that noise is not the only factor that affects detection errors. The channel and filter in the receiver could also lead to a phase shift and thus increase the amount of detection errors. In the end, through a combination of textbook and practical work, such subtle differences could be answered.

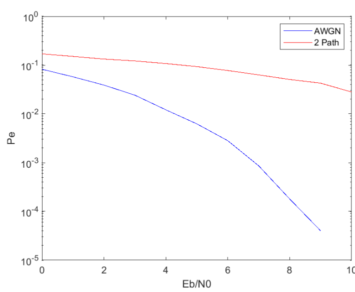


(a) The normal AWGN channel

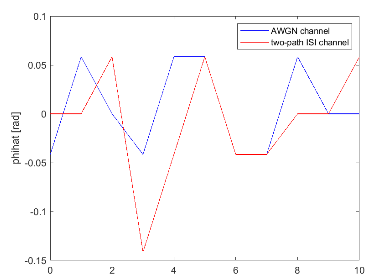


(b) The two-path ISI channel

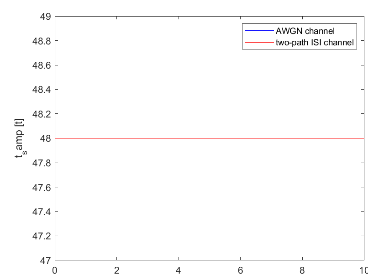
Figure 4: The eye diagram for different channels and an $E_b/N_0 = 10$ dB



(a) BER of two-path ISI and AWGN channel

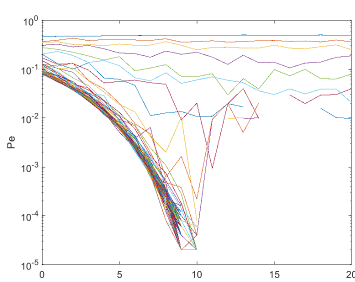


(b) Phase estimation accuracy

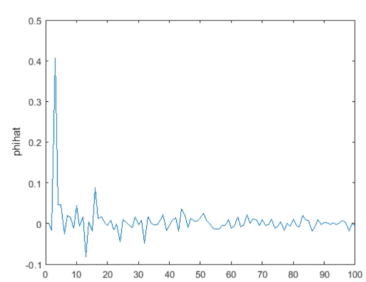


(c) Synchronization accuracy

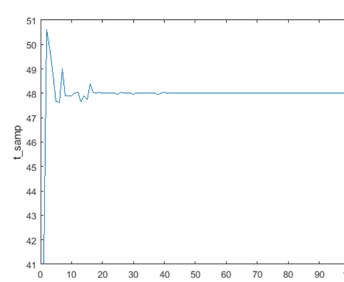
Figure 5



(a) BER for a training length of 0 to 100



(b) Phase estimate accuracy for a training length of 0 to 100



(c) t_samp for a training length of 0 to 100

Figure 6

A Appendix

All the code used in this project can be found on this GitHub repository:
`https://github.com/gomalodon/EQ2310_project`

Furthermore here are the requested functions (also available in the GitHub repository)

A.1 qpsk.m

```
function d = qpsk(b)
% d = qpsk(b)
%
% Map the bits to be transmitted into QPSK symbols using Gray coding. The
% resulting QPSK symbol is complex-valued, where one of the two bits in each
% QPSK symbol affects the real part (I channel) of the symbol and the other
% bit the imaginary part (Q channel). Each part is subsequently PAM
% modulated to form the complex-valued QPSK symbol. The energy per QPSK
% symbol is normalized to unity.
%
% The mapping resulting from the two PAM branches are:
%
% complex part (Q channel)
%         ^
%         |
%  10 x   |   x 00   (odd bit, even bit)
%         |
%  -------+------->  real part (I channel)
%         |
%  11 x   |   x 01
%         |
%
%
%
% Input:
%   b = bits {0, 1} to be mapped into QPSK symbols
%
% Output:
%   d = complex-valued QPSK symbols
n = size(b,2);
d = zeros(1,floor(n/2));
k = 1;
for i=1:n
    if mod(i,2) == 1
        if b(i) == 0
            d(k) = 1;
        else
            d(k) = -1;
        end
    else
        if b(i) == 0
            d(k) = d(k) + 1j;
        else
            d(k) = d(k) + -1j;
        end
        d(k) = d(k)/norm(d(k));
        k = k + 1;
    end
end
```

A.2 detect.m

```
function bhat = detect(r)
% bhat = detect(r)
%
% Computes the received bits given a received sequence of (phase-corrected)
% QPSK symbols. Gray coding of the individual bits is assumed. Hence, the
% two bits for each symbol can be detected from the real and imaginary
% parts, respectively. The first of the two bits below is output first in
```

```
% the bhat-sequence.
%
% Assumed mapping:
%
%  10 x   |   x 00
%         |
%  -------+-------
%         |
%  11 x   |   x 01
%
% Input:
%   r  = sequence of complex-valued QPSK symbols
%
% Output:
%   bhat  = bits {0,1} corresponding to the QPSK symbols
n = length(r);
bhat = false(1,2*n);
for i=1:n
    if real(r(i)) > 0
        bhat(2*i-1) = 0;
    else
        bhat(2*i-1) = 1;
    end
    if imag(r(i)) > 0
        bhat(2*i) = 0;
    else
        bhat(2*i) = 1;
    end
end
```

A.3 phase_estimation.m

```
function phihat = phase_estimation(r, b_train)
% phihat = phase_estimation(r, b_train)
%
% Phase estimator using the training sequence. The phase estimate is
% obtained by minimizing the norm of the difference between the known
% transmitted QPSK-modulated training sequence and the received training
% part. NB! There are other ways of estimating the phase, this is just
% one example.
%
% Input:
%   r       = received baseband signal
%   b_train = the training sequence bits
%
% Output:
%   phihat     = estimated phase
phihat = 0;
qpsk_train = qpsk(b_train);
n = r(1:length(qpsk_train));
min_g = norm(n-qpsk_train);
% Define the range and the step of sequence
for phase = -pi:0.1:pi
    r_sym = n*exp(-1j*phase);
    min_l = norm(r_sym-qpsk_train);
    if min_l < min_g
        min_g = min_l;
        phihat = phase;
    end
end
```

A.4 sync.m

```
function t_samp = sync(mf, b_train, Q, t_start, t_end)
% t_samp = sync(mf, b_train, Q, t_start, t_end)
%
% Determines when to sample the matched filter outputs. The synchronization
% algorithm is based on cross-correlating a replica of the (known)
% transmitted training sequence with the output from the matched filter
% (before sampling). Different starting points in the matched filter output
```

```matlab
% are tried and the shift yielding the largest value of the absolute value
% of the cross-correlation is chosen as the sampling instant for the first
% symbol.
%
% Input:
%   mf            = matched filter output, complex baseband, I+jQ
%   b_train       = the training sequence bits
%   Q             = number of samples per symbol
%   t_start       = start of search window
%   t_end         = end of search window
%
% Output:
%   t_samp = sampling instant for first symbol
L = t_end - t_start;
c = qpsk(b_train);
correlations = zeros(1,L);
for i=t_start:t_end
    temp = mf(i+((0:length(c)-1).*Q));
    correlations(i - t_start + 1) = abs(temp*c');
end
[top,index] = max(correlations);
t_samp = index + t_start - 1;

if false
    stem(t_start:t_end,correlations, "b")
    hold on
    stem(t_samp,top, "r")
end
```