The Q Function and Baseband Data Communication

Eren Can Gungor
Riccardo Miccini
Technical University of Denmark - DTU

December 14, 2016

Contents

1 Eye Diagram for a Digital Communication Channel

1.1 Eye diagram

A so-called *eye diagram* is the pattern that originates from overlapping a digital signal over the length of one or more transmitted bits. For several types of digital modulation the plot would show a series of round shapes (eyes) delimited by two rails, hence the name. The rails are formed by the low-frequency components derived by the bit sequences 00 and 11, whereas the round shapes are formed by the rising and falling edges of the high-frequency bit sequences 01 and 10.

This tool is commonly used to investigate several performance measures of a transmission channel, including noise level, distortion, inter-symbol interference, and synchronization errors.

An eye diagram can be generated with an oscilloscope by enabling infinite persistence, and setting the trigger to react on a separate clock signal.

1.2 `c5ce2.m`: explanation

Here follows a thoroughly commented version of the provided `c5ce2.m` MATLAB script. The code below generates and plots the eye diagrams of four band-limited signals composed of random sequences of bits.

Listing 1: ../scripts/1/c5ce2.m

```
% clean figure and load signal package (only for Octave)
clf
pkg load signal

% simulation parameters:
%  - nr of symbols (must be divisible by 4)
%  - nr of samples per symbol
%  - filter cutoff values (normalized bandwidth)
nsym = 1000;
nsamp = 50;
bw = [0.4 0.6 1 2];

% for each filter cutoff value..
for k = 1:length(bw)
  % generate filter coefficients using one of the cutoff values
  lambda = bw(k);
  [b,a] = butter(3,2*lambda/nsamp);

  % allocate space for total bit sequence
```

```
  l = nsym*nsamp;
  y = zeros(1,l-nsamp+1);

  % initalize random output vector with +1 and -1
  x = 2*round(rand(1,nsym))-1;

  % for each overlap..
  for i = 1:nsym
    % place one symbol into vector y, leaving nsamp samples of
       spacing
    kk = (i-1)*nsamp+1;
    y(kk) = x(i);
  end
  % zero-order hold
  datavector = conv(y,ones(1,nsamp));

  % apply filter to the total sequence
  filtout = filter(b, a, datavector);

  % splice sequence into sub-sequences of 4 symbols
  datamatrix = reshape(filtout, 4*nsamp, nsym/4);

  % discart the first 5 sub-sequences
  datamatrix1 = datamatrix(:, 6:(nsym/4));

  % plot eye diagram
  subplot(length(bw),1,k)
  plot(datamatrix1, 'k')

  % format: print axis labels, legend, and set plot range
  ylabel('Amplitude')
  axis([0 4*nsamp -1.4 1.4])
  legend(['Bn = ', num2str(lambda)])
  if k == 4
    xlabel('t/Tsamp')
  end
end
```

1.3 Channel model

In order to analyze the performances of a given transmission system, a model of the adopted transmission channel has to be devised.

For the purpose of this simulation, the most interesting characteristic of the channel is its bandwidth, and therefore it suitability for transmitting information at a specific rate. The limitation in bandwidth is obtained by feeding the signal into a third-order Butterworth low-pass filter.

The channel is therefore characterized by its *normalized bandwidth*, which represents the bandwidth as a percentage of the data bit rate; e.g if the bit rate is 8000Hz and the normalized bandwidth is 0.5, the channel will have a cutoff frequency of 4000Hz.

The MATLAB tools for generating digital filters (`butter`, `ellip`, `cheby1`...) accepts the normalized frequency as input argument, which is the cutoff frequency in terms of the *Nyquist frequency* (half of the sampling frequency). The normalized bandwidth is proportional to the normalized frequency by a factor equivalent to the number of samples per bit.

1.4 `c5ce2.m`: different bandwidths

The following section will present a modified version of the previously introduced script, which can plot the eye diagrams for the normalized bandwidth 0.15, 0.3, 0.7, 1.2, and 4.

In order to do so, the input vector of `bw` has been changed to reflect the given values, together with the subplot format, and x-axis label clause.

Listing 2: ../scripts/1/c5ce2_augm.m

```
% clean figure and load signal package (only for Octave)
clf
pkg load signal

% simulation parameters:
%  - nr of symbols (must be divisible by 4)
%  - nr of samples per symbol
%  - filter cutoff values (normalized bandwidth)
nsym = (100)*4;
nsamp = 50;
bw = [0.15 0.3 0.7 1.2 4];

% for each filter cutoff value..
for k = 1:length(bw)
  % generate filter coefficients using one of the cutoff values
  lambda = bw(k);
  [b,a] = butter(3,2*lambda/nsamp);

  % allocate space for total bit sequence
  l = nsym*nsamp;
  y = zeros(1,l-nsamp+1);

  % initalize random output vector with +1 and -1
  x = 2*round(rand(1,nsym))-1;

  % for each overlap..
```

```
  for i = 1:nsym
    % place one symbol into vector y, leaving nsamp samples of
       spacing
    kk = (i-1)*nsamp+1;
    y(kk) = x(i);
  end
  % zero-order hold
  datavector = conv(y,ones(1,nsamp));

  % apply filter to the total sequence
  filtout = filter(b, a, datavector);

  % splice sequence into sub-sequences of 4 symbols
  datamatrix = reshape(filtout, 4*nsamp, nsym/4);

  % discart the first 5 sub-sequences
  datamatrix1 = datamatrix(:, 6:(nsym/4));

  % plot eye diagram
  subplot(length(bw),1,k)
  plot(datamatrix1, 'k')

  % format: print axis labels, legend, and set plot range
  ylabel('Amplitude')
  axis([0 4*nsamp -1.4 1.4])
  legend(['Bn = ', num2str(lambda)])
  if k == length(bw)
    xlabel('t/Tsamp')
  end
end
```

In order to properly visualize the resulting plots, the original bandwidth values have been removed. In order to view all the eight eye diagrams, line 11 can be substituted with the following:

Listing 3:

```
bw = [0.15 0.3 0.7 1.2 4];
```

1.5 `c5ce2.m`: plots

This section will elaborate on the structure and implications of the eye diagrams generated using the scripts above.

The plots are generated by the original and augmented versions of the script respectively, and show the eye diagram for an antipodal baseband transmission at various channel bandwidths.

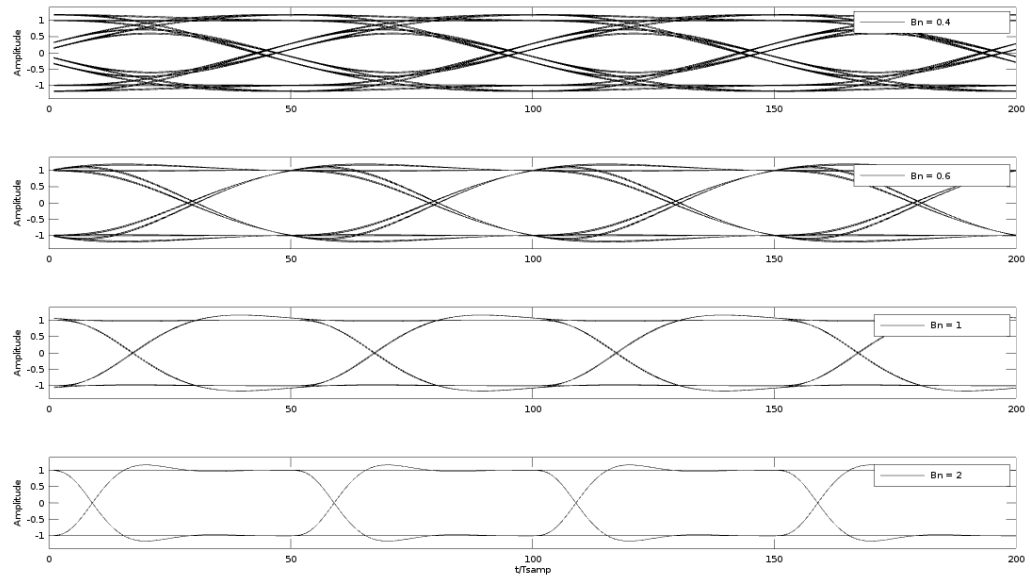

Figure 1: Eye diagrams of signals with filter coefficients 0.4, 0.6, 1, and 2

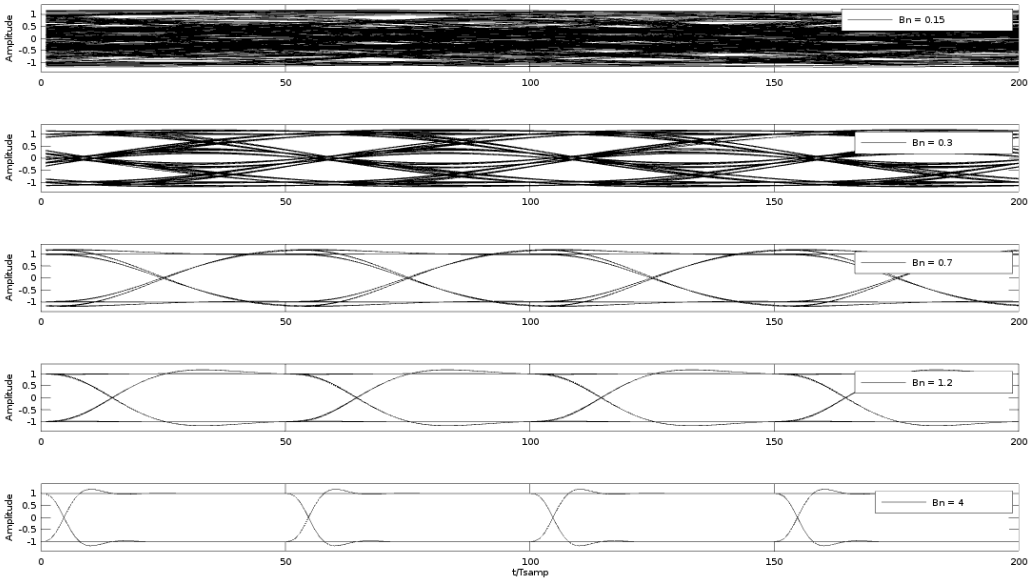

Figure 2: Eye diagrams of signals with filter coefficients 0.15, 0.3, 0.7, 1.2, and 4

When observing an eye diagram, the two main measures of the system performance are the amplitude jitter A_j and the timing jitter T_j.

The former measures the difference in amplitude between the low-frequency and high-frequency components of the signals, sampled where the ye is at its maximum opening. Low channel bandwidth causes the eye to shrink and amplitude jitter to increase, which in turns increases the bit error probability.

The latter parameter represents the time interval at which the zero-crossing may occur. It is caused by inter-symbol interference, and a high value could result in synchronization issues.

The eye diagrams from both scripts clearly show how a low bandwidth results in overall more jittery and less intelligible signals. For the normalized bandwidth value of 0.15 and 0.3,

2 The Q function

2.1 Normal probability density function 2.1

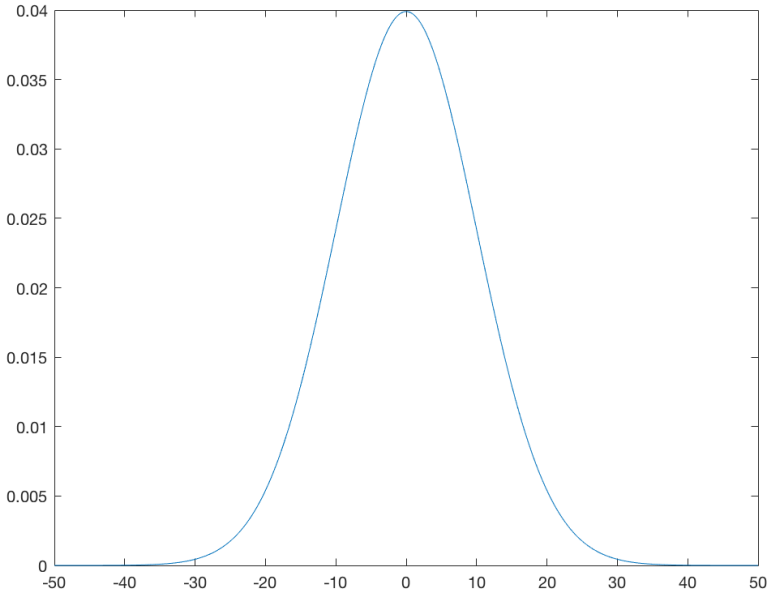

Figure 3: Normal Gaussian pdf graph with defined intervals

Normal gaussian distribution is one of the most important concept in Communication system and in statistics. It is used as a powerful tool when investigating random signals in communication systems. Such as investigating behaviour and application of noise signals.

The other reason is that because it is characterised by the limited variables. Such as mean μ and σ. It is easier compute and understand when we apply on communication systems.

All the variables that has been used in graphingpdf.m has been explained below;

mu . This is the mean value (μ) for the normal probability density function.

sigma This the sensible standard deviation number. (σ).

MAX 50; Maximum x value that x vector will get

MIN -50; Minimum x value that x vector will get

Also general formula for gaussian pdf is ; y=$f(x|\mu,\sigma) = (\frac{1}{\sigma\sqrt{2\pi}})e^{\frac{-(x-\mu)^2}{2(\mu)^2}}$
As we can see and understand from the variables above and the formula that all variables has been specified and we only need σ, μ and range.

2.2 Explanation of Q(u) function in relation to the normal probability density function 2.2

As we have explained and investigated probability density density function above . We can easily link $Q(u)$ function by exploiting the properties of cumulative distribution function and Probability density function.

- Q function is the 1- minus the cumulative distribution function of the standardised normal variable.
- Gaussian pdf with unit variance and zero mean is R=$(\frac{1}{\sqrt{2\pi}})e^{\frac{-(x)^2}{2}}$
- And corresponding cumulative distribution function become; $P = \int_{-\inf}^{x} Z$
- In last step Gaussian Q function defined as $Q(x) = 1 - P(x) = \int_{x}^{-\inf} Z$

2.3 Preparing a script file for plotting the $Q(u)$ function for argument values of relevance to the detection problems for digital communication receivers and inserting them in Appendix 2.3 and 2.4

Script file that has been created for the assignment 2.3 and 2.4 has been added to the Appendix. Important concept has been explained and relevant digital communication input values specified.

2.4 2.5

2.4.1 Complementary error function

2.5 Plots

Plotting for the questions 2.3 and 2.4

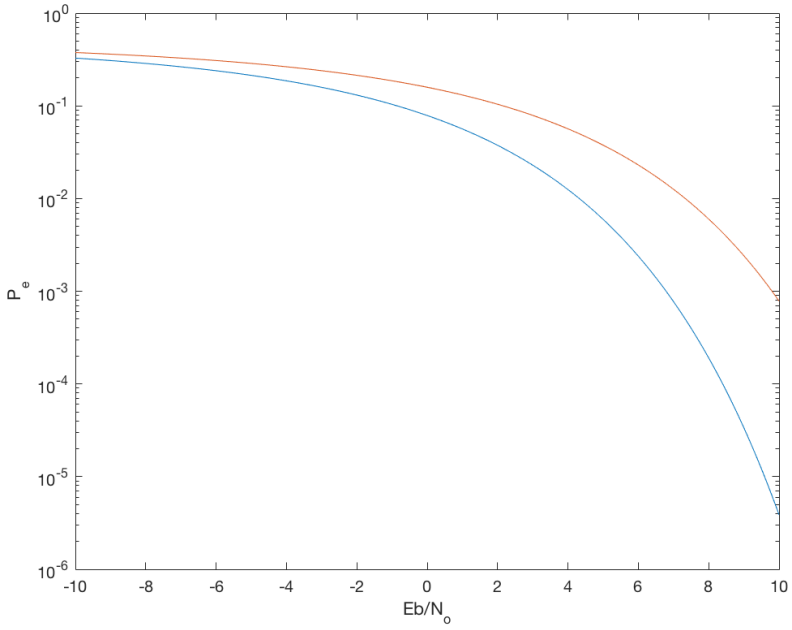

Figure 4: Plot for the $Q(u)$ function relevant to the argument values for the detection problems for digital communication receivers

3 Source Code

Here we have the properly prepared MATLAB codes for the second part of the second project. It has been used for observations, calculations and comparing with specified commands that given in this project.

The code belows computes and graphs normal(Gaussian) probability density function (pdf) in an appropriate intervals

Listing 4: ../scripts/2/graphingpdf.m

```
%graphing PDF function with random variables
mu=0;
sigma=10;
MAX = 50;
MIN = -50;
STEP = (MAX - MIN) / 1000;
PDF = normpdf(MIN:STEP:MAX, mu, sigma);
plot(MIN:STEP:MAX, PDF)
```

For the question 2.3 and 2.4 we have created the following code below. This code will plot the $Q(u)$ function for argument values relevant to the

digital communication receivers. As we know from previous chapters that there are two common argument values :

- $R_1 2 = 0$ (Orthogonal Signals)
- $R_1 2 = -1$ (Antipodal Signals)

We also used $z_d b$ for the ratio for E_b/N_o. We should also notice that $z_d b$ is dimensionless ratio.

After that we have investigated the graph for further investigation to see whether we have achieved a satisfactory results for the assignment 2.3 and 2.4. Mathematical calculations are matching up with MATLAB simulation results.

Listing 5: ../scripts/2/qfunction.m

```
z_db = -10:.1:10;
r12 = [-1 0];
z = db2pow(z_db);
p_e(1,:) = qfunc(sqrt((1 - r12(1))*z));
p_e(2,:) = qfunc(sqrt((1 - r12(2))*z));
semilogy(z_db, p_e(1,:))
hold on
semilogy(z_db, p_e(2,:))
xlabel('Eb/N_o')
ylabel('P_e')
```

4 The Matched Filter Base Band Receiver

4.1 Explain the principles of a matched filter receiver for binary data in white Gaussian noise, thus using the so called AWGN (Additive White Gaussian Noise) model. The description should include a definition of all the necessary variables in the system and a block diagram 3.1

Firstly, we will use the block diagram below to explain important variables for the matched filter receiver for binary data in AWGN.

A matched filteris achieved bycorrelatinga knownsignal(reference signal) with an unknown signal to detect the presence of the template in the unknown signal. The matched filter is the "*optimal*" linear filter for maximising the SNR(signal-to-noise ratio) in the presence of additive stochastic noise. In this specific case, our additive noise is white gaussian noise.

The matched filter with AWGN is used in communication systems that sends binary messages from transmitter to receiver through AWGN channel. It is important to firstly define error analysis for general binary noise signals as we try to optimise signal to noise ratio for the filter.

For general error analysis for binary signals in noise are;

- Receive 0 sends 1
- Receive 1 sends 0

After we received and processed binary signals. The received signal is filtered and filter output is sampled every T seconds. As we can see from the figure that threshold will decide which signal it will pass with respect to gain k.

To understand this concept further from error analysis following important variables has been considered. Those are

- Minimize the average probability error
- Choose the optimal threshold
  - Our optimal threshold formula is $0.5[s_1(T) + s_2(T)]$

As we know that P_E is a function of the difference between two signals.Therefore , P_E will decrease with increasing argument values. To achieve that we need to make $h(t)$ such that P_E is a minimum when difference between the signals at the output of the filter is maximum.

Solution for that would be $h_0(t) = s_2(T-t) - s_1(T-t)$ As we can see that the optimum filter related to only the input signals.

Important variables has been derived below to see how important they are for the system and enhance our understanding of the block diagram . We will drive formulas for optimum filter and error probability.

4.1.1 Derivation for Error Probability for Matched Filter

Recall $P_E = Q(d/2)$ and we know that our maximum value of the distance is:

- $d_m = \frac{2}{N_o}(E_1 + E_2 - 2\sqrt{E_1 E_2}\rho_1 2)$

We know the energy formulas from the previous chapters. These energy formulas can be used to define ρ formula. Following steps has been done to represent $\rho_1 2$ in terms of energy.

- $E_1 = \int_{t_o}^{t_0+T} s_1^2(t)dt$
- $E_2 = \int_{t_o}^{t_o+T} s_2^2(t)dt$
- $\rho_1 2 = \frac{1}{\sqrt{E_1 E_2}} \int_{-\infty}^{\infty} s_1(t)s_2(t)dt$

And finalised version of Probability error will become:

- $P_E = Q((\frac{E_1+E_2-2\sqrt{E_1E_2}\rho_1 2}{2N_o})^{1/2})$

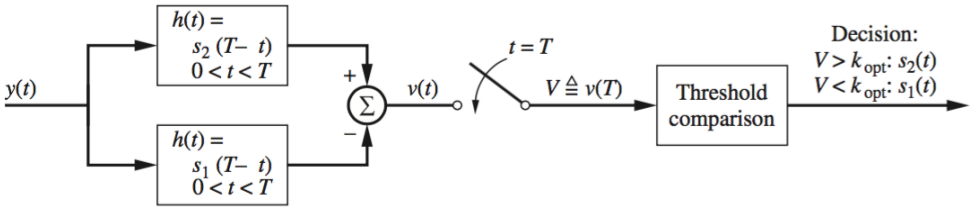

Figure 5: Matched Filter for binary data in AWGN

4.2 Explain the code in `c8ce1a.m` in greater detail than the present comments. Insert the comments directly in a renamed version of the script 3.2

Here follows a thoroughly commented version of the provided `question32.m` MATLAB script. The code below generates and calculates gain estimate,delay

estimate, estimate of px, estimate of py and snr estimate in ratios by given reference and measurement sine waves.

It can be seen below:

Listing 6: ../scripts/2/question32.m

```
% File: c8ce1.m
t = 1:6400; % time interval that given between 1 to 6400
fs = 1/32;  % Sampling Frequency that has been specified for
    this assignment
x = 2*sin(2*pi*fs*t); % Reference signal that consists of a sine
     wave
y = 10*sin(2*pi*fs*t+pi)+0.1*sin(2*pi*fs*10*t); % Measurement
    signal that has been specified for this argument. One has a
    sligthly higher frequency component.
[gain,delay,px,py,rxymax,rho,snrdb] = snrest(x,y); % This snrest
     estimates the given output arguments
snr=db2pow(snrdb); % Convert signal to noise ratio from decibels
     to a ratio
format long e % it will set the output format to the long fixed-
    decimal format and display value of e
a = ['The gain estimate is ',num2str(gain),'.']; % it will
    convert gain estimate number into a string
b = ['The delay estimate is ',num2str(delay),' samples.'];% it
    will conver delay estimate number  into a string
c = ['The estimate of px is ',num2str(px),'.']; % it will
    convert power in reference vector number estimate into a
    string
d = ['The estimate of py is ',num2str(py),'.']; % it will
    convert gain estimate number into a string
e = ['The snr estimate is ',num2str(snr),'.']; % it will convert
     signal to noise ratio number convert into a string
f = ['The snr estimate is ',num2str(snrdb),' db.']; % it will
    convert snr estimate in db into a string
disp(a); disp(b); disp(c); disp(d); disp(e); disp(f) % it will
    display the values a,b,c,d in a string at output
%px= determines power in reference vector
%py=determine power in measurement vector
%rxymax= finds the max of the cross-correlation matrix
%rho=estimate of the correlation coefficient
%snr= estimate of signal to noise ratio
%snrdb= snr estimate in db
```

4.3 3.3

4.4 `c9ce1a.m`: Extend the code in c9ce1a.m such that the number of input correlations can be up to 8. Presently 4 is the maximum of input arguments 3.4

For the part 3.4 `question35.m` MATLAB code has been created. It can be seen that code is able take 8 correlation input. It is accessible from the appendix part of this report.

4.5 Apply the following correlation coefficients in the c9ce1a script $[-1 -0.75 -0.50 0.5 0.75 0.8 .995]$ and insert the resulting plot in the project report. Explain the plot and verify that it is correct 3.5

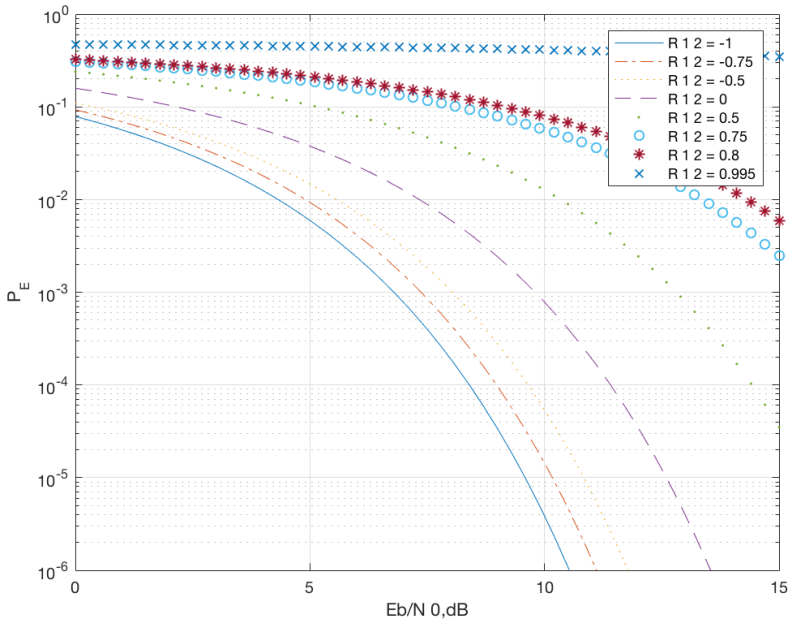

Figure 6: P_E over E_b/N_o graph for investigating how it will change with introduced correlation coefficients

4.6 Explain how a matched-filter receiver can be implemented by a correlator. The explanation should include the most important formulas leading to the matched filter implementation using a cross correlation Part 3.6

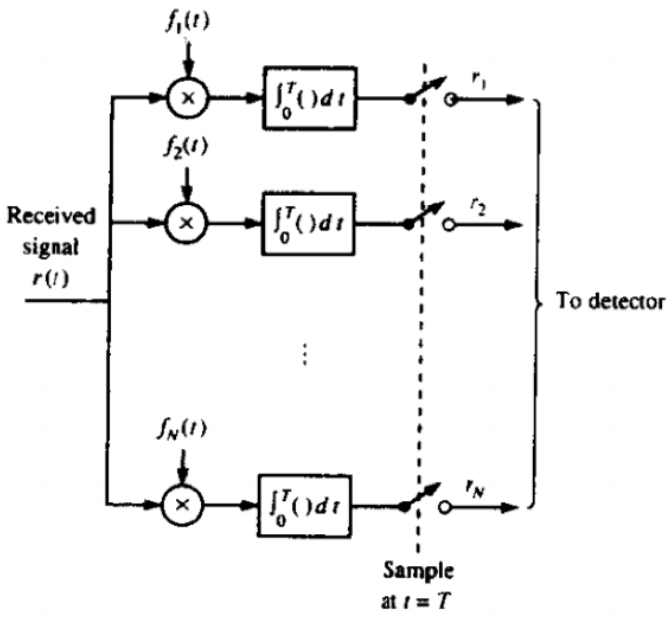

Figure 7: Correlator Matched Filter

As we can from the Figure 3 that we can implement the matched filter by implementing the correlation detectors. The cross-correlator does the cross-correlation between the noisy signal and noiseless signal.

If we look at the matched filter, it does the convolution between the received signal and the time-reversed copy of the reference signal. We can mathematically prove that both matched filter and the cross-correlator will give us same output signal values. To prove this we will derive mathematical derivation.

Suppose we have received $r(t)$ and is passed through parallel bank of N ?crosscorrelators? which basically compute the prediction of $r(t)$ onto the N-basis functions as $f_n(t)$.This can be seen in Figure 3.

For cross-correlator signal is now represented by $s_m(t)$ with components $s_{mk}(t)$. $s_{mk}(t)$ values depend on how many M_{signal} is transmitted. Now other

components of cross-correlator will be explained below for the mathematical derivation for the output signal.

- $\int_0^T r(t)f_k(t) = \int_0^T [s_m(t) + n(t)]f_k(t)dt$
- $r_k(t) = s_m(t) + n_k(t)$
- $\int_0^T s_m(t)f_k(t)$ for the values of $k = 1, 2, 3, 4, 5, ...$
- $\int_0^T n(t)f_k(t)$ for the values of $k = 1, 2, 3, 4, 5.....$

Now our signal has been represented by the $s_m(t)$ and the $n(k)$ (random variables as noise)

Now the signal can be represent $r(t)$ as

- $r(t) = \sum_{k=1}^{N} = s_m(t)f_k(t) + n(t)' + \sum_{k=1}^{N} = n_k(t)f_k(t)$
- $\sum_{k=1}^{N} = r_k(t)f_k(t)$

As we can see that $n(t)'$ is irrelevant to which signal is going to transmitted. The decision will be made upon entirely on correlator output and the basis functions.

Now we can compare with Matched filter output.

For matched filter, we use N bank linear filters. Impulse responses of the N filters are;

- $h_k(t) = f_k(T - t)$ for invertals between , $0 < t < T$

The output filters become:

- $y_k(t) = \int_0^T r(\tau)f_k(T - \tau)d(\tau)$
- $\int_0^T r(\tau)f_k(T - t + \tau)d(\tau)$ for the $k = 1, 2, 3, 4, 5, ...N$

If we sample outputs of the linears at $t = T$

- $y_k(T) = \int_0^T r(\tau)f_k(\tau)d(\tau)$ which will eventually become r_k

Hence our theoritical approach has been proved by mathematical derivation of matched filter and cross-correlator. Both have same output even though calculations have been done in a different way for both.

4.7 Explain what the consequences are with respect to a matched-filter receiver if there is noise on the timing synchronisation in the receiver Part 3.7

Even if we manage to recover the timing, it does not guarantee that the correct operation of data-aided frequency estimation algorithms. The reason is that for is the presence of noise on the timing synchronisation in the receiver. Due to fact that for a frequency offset in order of $1/T$ the signal will be severely distorted when it passed through the matched filter. Severely distorted matched filter will not able to maximise the signal to noise ratio in the presence of additive noise.

4.7.1 Example for the Part 3.7

For example, if our noise signal is delta, sampling the matched filter?s output at some time $T+\delta$ (where represents a receiver timing offset due to introduction of noise) will significantly reduce the effective SNR seen by subsequent receiver blocks. This example that linked to theory above proves that it is important to keeping receiver timing offset close to zero as possible and thus delivers motivation for the inclusion of a timing recovery loop in the receiver.

4.8 Plots

4.9 Source Code

For the question 3.5 following code has been created to take relevant correlation coefficients

Listing 7: ../scripts/2/question35.m

```
% file: c9ce1
% Bit error probability for binary signaling;
% vector of correlation coefficients allowed
%
clf
R12 = input('Enter␣vector␣of␣desired␣R␣1␣2␣values;␣<=␣8␣values␣'
    );
A = char('-','-.',':','--','.','o','*','x');
LR = length(R12);
z_dB = 0:.3:15;
z = db2pow(z_dB);
for k = 1:LR
    P_E=qfn(sqrt(z*(1-R12(k)))); % Probability of error for
        vector of z-values
```

```
                                % Plot probability of error
                                   versus Eb/N0 in dB
    semilogy(z_dB,P_E,A(k,:)),axis([0 15 10^(-6) 1]),xlabel('Eb/
       N␣0,dB'),ylabel('P_E'),...
    if k==1
    hold on; grid % Hold plot for plots for other values of R12
       end
    end
end
if LR == 1 % Plot legends for R12 values
legend(['R␣1␣2␣=␣',num2str(R12(1))])
elseif LR == 2
legend(['R␣1␣2␣=␣',num2str(R12(1))],['R␣1␣2␣=␣',num2str(R12(2))
   ])
elseif LR == 3
legend(['R␣1␣2␣=␣',num2str(R12(1))],['R␣1␣2␣=␣',num2str(R12(2))
   ],['R␣1␣2␣=␣',num2str(R12(3))])
elseif LR == 4
legend(['R␣1␣2␣=␣',num2str(R12(1))],['R␣1␣2␣=␣',num2str(R12(2))
   ],['R␣1␣2␣=␣',num2str(R12(3))],['R␣1␣2␣=␣',num2str(R12(4))])
elseif LR == 5
legend(['R␣1␣2␣=␣',num2str(R12(1))],['R␣1␣2␣=␣',num2str(R12(2))
   ],['R␣1␣2␣=␣',num2str(R12(3))],['R␣1␣2␣=␣',num2str(R12(4))],[
   'R␣1␣2␣=␣',num2str(R12(5))])
elseif LR == 6
legend(['R␣1␣2␣=␣',num2str(R12(1))],['R␣1␣2␣=␣',num2str(R12(2))
   ],['R␣1␣2␣=␣',num2str(R12(3))],['R␣1␣2␣=␣',num2str(R12(4))],[
   'R␣1␣2␣=␣',num2str(R12(5))],['R␣1␣2␣=␣',num2str(R12(6))])
elseif LR == 7
legend(['R␣1␣2␣=␣',num2str(R12(1))],['R␣1␣2␣=␣',num2str(R12(2))
   ],['R␣1␣2␣=␣',num2str(R12(3))],['R␣1␣2␣=␣',num2str(R12(4))],[
   'R␣1␣2␣=␣',num2str(R12(5))],['R␣1␣2␣=␣',num2str(R12(6))],['R␣
   1␣2␣=␣',num2str(R12(7))])
elseif LR == 8
legend(['R␣1␣2␣=␣',num2str(R12(1))],['R␣1␣2␣=␣',num2str(R12(2))
   ],['R␣1␣2␣=␣',num2str(R12(3))],['R␣1␣2␣=␣',num2str(R12(4))],[
   'R␣1␣2␣=␣',num2str(R12(5))],['R␣1␣2␣=␣',num2str(R12(6))],['R␣
   1␣2␣=␣',num2str(R12(7))],['R␣1␣2␣=␣',num2str(R12(8))])

end
```