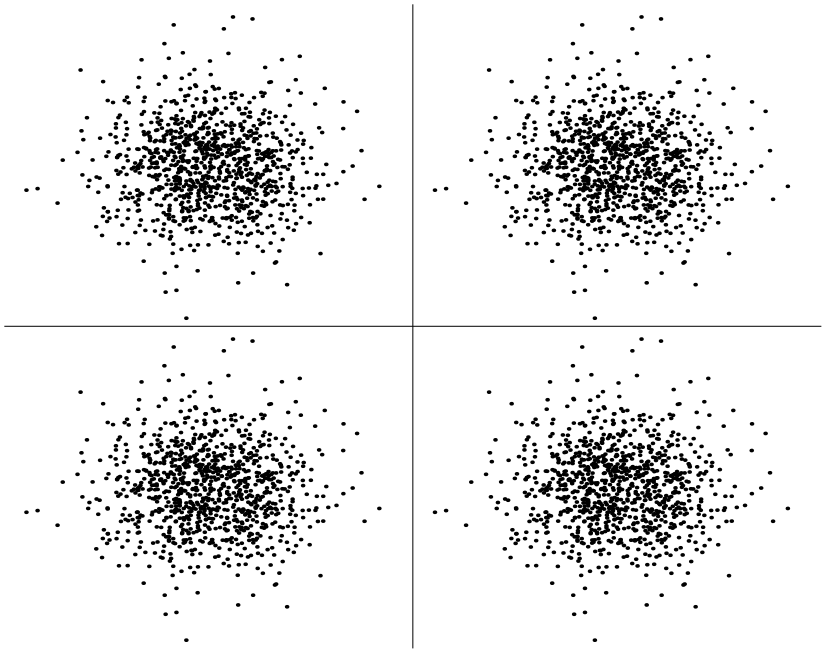
Project Assignment
EQ2310 Digital Communications

Analysis and Simulation of a QPSK System



November 11, 2015

Project Assignment

This document specifies a project assignment used in the course EQ2310 Digital Communications.

General Information 2015

Read the whole text carefully before you start working on the assignment. Note that your work has to be reported in a separate technical report. The reports will be graded passed or failed. Section 6 lists some requirements that have to be fulfilled. Note, in particular, that

- this project assignment is a mandatory part of the course and is worth 1 credit unit.
- you are allowed to cooperate in groups of *at most two* students (but solving the assignment individually is also perfectly OK).
- two students that have cooperated may choose to write one report each or one common report, with two authors. If you write separate reports, but have cooperated, then state who you have worked with (see the marking-template and below).
- the report should be neatly structured, easy to follow and is not allowed to exceed six pages (excluding the cover page and the Matlab code; see below). Unreadable reports will be failed immediately.
- use a word-processor of your choice to write the report (hand-written reports will not be acceptable).
- the language of the report can be either Swedish or English.
- a listing of the Matlab code written by the student(s) as part of the assignment must be enclosed as a separate appendix to the report.
- the *template included as last page* of this document will be used for grading. *Use this page as cover page when you hand in the report.*

The **deadline for handing in the report is Friday, December 4, 2015** (hand in at STEX, or email to lkra@kth.se). Reports handed in too late will not be corrected.

After having been corrected, the reports can be checked out at STEX. There will be a second deadline for revising failed reports. This deadline will be announced when all reports have been checked and made available for potential revision. (The second deadline, as well as any further information on this topic, will be posted on the course homepage).

Note that reports that are failed *a second time* will *not* be considered for a new revision until the next time the course is given (Fall of 2016/17).

1 Introduction

The purpose of this exercise, which can be considered being a "hybrid" between a traditional laboratory experiment and a homework problem, is to illustrate some important aspects on analysis and simulations, both baseband and passband, of communication systems. Aspects such as bit error probability, synchronization, carrier phase estimation, eye diagram, spectral properties, and practical imperfections due to filtering will be studied in a QPSK system operating over an AWGN channel.

The outcome of the exercise, in addition to increasing your knowledge of communication theory, is a short written technical report were you discuss your findings and comment on various aspects you find interesting. No list of questions is provided, but instead some general areas to be discussed in the report are provided. Thus, there are great opportunities to concentrate on those areas that interest you the most.

All the simulations will be carried out using Matlab and some of the software has to be written by yourself as only parts of the transmitter and a skeleton of the receiver is provided.

The current report starts with a brief description of passband and baseband signals in Section 2. In Section 3, the technique used for simulating a time-continuous system using a time-discrete computer is outlined. The treatment in these two sections contains quite a lot of mathematics, but all the relations are known from previous courses. It can also be a good idea to read the whole report even if there are some details you do not understand yet and then return to the first two sections once more. Section 4 outlines the system that will be simulated. The focus is on the functions carried out by each block and not on the mathematical details. In Section 5, the Matlab functions used for simulating the system, as well as the functions that are missing and have to be supplied by yourself, are described. Finally, the requirements on the technical report, which is the result of your work, are outlined in Section 6.

2 Baseband Representation

In many communication systems the baseband signal that conveys the message to be transmitted is up-converted (i.e., translated in frequency) in order to better suit the characteristics of the channel. An example of such a system is a QPSK system. The QPSK modulation process can be viewed as a two step procedure. First, a baseband signal, consisting of a series of complex valued pulses, is formed. This signal is then up-converted to the desired carrier frequency. The result is a passband signal which can be transmitted over a physical channel.

The goal of this section is to show how such a passband system can be given an equivalent baseband representation. It is our intention to provide an informal development of the subject.

Typically, simulation of a passband system is performed using the baseband equivalent system. The baseband equivalent system has a number of useful properties which makes it an attractive choice for simulations. A more detailed discussion of the advantages with a baseband representation is found in Section 2.2.

2.1 Baseband vs Passband

This section provides a quick review of the concept of baseband equivalent signals. In general, any amplitude/phase modulation technique can be described by the relation

$$x(t) = A(t)\cos(2\pi f_c t + \Phi(t)) \ , \tag{1}$$

where f_c is the carrier frequency. The bandwidths of the phase function $\Phi(t)$ and the amplitude function $A(t)$ are in general much lower than the carrier frequency. Hence, the rate-of-change in these signals is typically much lower than f_c. Consequently, $x(t)$ is a passband signal with its spectrum concentrated around the carrier frequency. Using trigonometric relations, it is possible to write the same function as

$$x(t) = x_{\rm I}(t)\cos(2\pi f_c t) - x_{\rm Q}(t)\sin(2\pi f_c t) \ , \tag{2}$$

where

$$x_{\rm I}(t) = A(t)\cos(\Phi(t)) \tag{3}$$
$$x_{\rm Q}(t) = A(t)\sin(\Phi(t)) \tag{4}$$

represent the *quadrature* components. Here, $x_{\rm I}(t)$ and $x_{\rm Q}(t)$ is the *in-phase* (I) and the *quadrature-phase* (Q) component, respectively. Equation (2) can be rewritten using complex numbers as

$$x(t) = {\rm Re}\{x_{\rm bb}(t)e^{j2\pi f_c t}\} \ , \tag{5}$$

where

$$x_{\rm bb}(t) = x_{\rm I}(t) + jx_{\rm Q}(t) \tag{6}$$

is the *baseband equivalent* signal.

It is instructive to study the passband and baseband equivalent signals in the frequency domain. As illustrated in Figure 1, the baseband equivalent signal is obtained from the passband signal by removing the image of the signal on the negative frequency axis, scaling the remaining spectrum a factor of two and moving the result to the baseband by shifting the spectrum f_c Hz to the left.

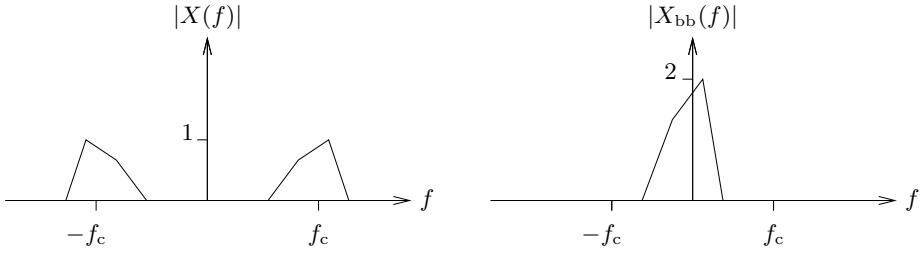


Figure 1: The relation between a passband signal and its baseband equivalent signal in the frequency domain.

A baseband equivalent signal can be obtained by using a down-converter. Such a device might be implemented in several ways. One example of an implementation is shown in Figure 2. The signal is first multiplied with $2e^{-j2\pi f_c t}$

in order to shift the spectrum and scale it by a factor of two. The low-pass (LP) filter then removes the negative frequency image. It is assumed that the low-pass filter is ideal and has a sufficiently large bandwidth so as not to alter the shape of the positive frequency image. A down-converter is found in most passband communication receivers.

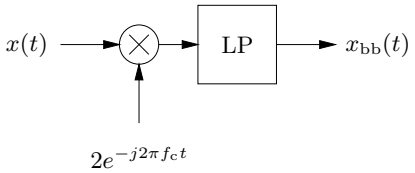


Figure 2: Converting a passband signal into its baseband equivalent signal.

Communication signals are usually represented using just the complex signal $x_{\mathrm{bb}}(t)$ in (6). This is called the *baseband equivalent* (baseband for short) representation as opposed to the *passband* representation $x(t)$, which is a real valued signal. The baseband representation is much easier to work with than the passband representation, as will be illustrated below.

2.2 Advantages of the Baseband Representation

The use of a baseband representation simplifies communications system simulation and analysis in a number of ways. First, only the slowly varying parts of the signal need to be processed. This saves considerable computing effort since the sampling rate used to simulate the continuous-time signals can be substantially lower, as will be made clearer in Section 3. Moreover, a simulation or analysis of a baseband system is not tied to any particular carrier frequency and can be reused if the carrier frequency is changed. Certain very useful operations become extremely simple when using the complex representation. For example, a frequency shift by f_{shift} is done by multiplying the signal by $e^{j2\pi f_{\mathrm{shift}} t}$. A phase shift by φ_{shift} is done by multiplying the signal by $e^{j\varphi_{\mathrm{shift}}}$.

2.3 A Baseband Equivalent Communication System

In this section, the relations between baseband and passband signals are utilized for converting a typical communication system with a carrier frequency to its baseband equivalent counterpart. As previously mentioned, this makes it considerably easier to simulate the system using a computer.

The communication system considered is shown in Figure 3. This is a typical setup which can represent any kind of system using quadrature amplitude modulation (QAM). The QPSK system in this report is one example.

A brief overview of the system now follows. At the transmitting side, the sequence of symbols $d(n)$ is converted to a continuous-time baseband signal $s_{\mathrm{bb}}(t)$ by a pulse amplitude modulator (PAM). Note that $d(n)$ takes values from a discrete set of *complex-valued* symbols. It is assumed that the signal constellation is normalized to unit average power, i.e. $\mathrm{E}[|d(n)|^2]=1$ and that the pulse function $p(t)$ is *real-valued* and *strictly bandlimited* to frequencies below W with $W < f_c$ (usually it is reasonable to assume $W \ll f_c$). Up-Conversion

is performed by multiplying with $e^{j2\pi f_c t}$ and keeping the real part, resulting in a passband signal $s(t)$ being transmitted over the channel (compare with (5)). The channel adds white Gaussian noise $w(t)$ with a spectral density $S_w(f) = N_0/2$. In order to remove the carrier, the received signal $r(t)$ is processed by a down-converter which outputs the corresponding baseband equivalent signal $r_{\rm bb}(t)$. The down-converter is followed by a symbol sampled matched filter and a detector which outputs estimates of the transmitted symbols.

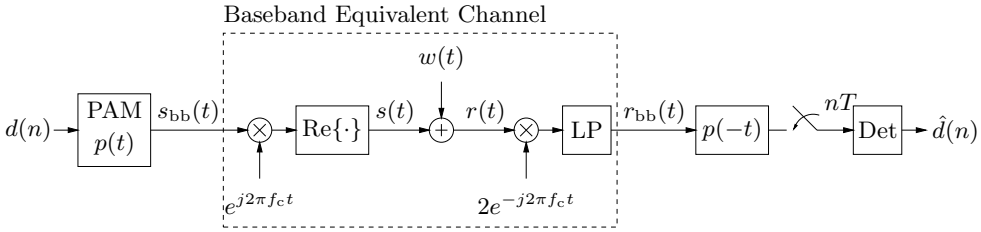


Figure 3: The original passband system.

An important parameter is the signal-to-noise-ratio (SNR). The SNR can be defined in many ways. Here it is defined as

$$\begin{aligned}\text{SNR} \triangleq \frac{E_{\rm b}}{N_0} = \frac{E_{\rm s}/k}{N_0} &= \frac{\text{E}[\|d(n)\text{Re}\{p(t)e^{j2\pi f_c t}\}\|^2]}{kN_0} \\ &= \frac{\text{E}[|d(n)|^2]\|\text{Re}\{p(t)e^{j2\pi f_c t}\}\|^2}{kN_0} \ , \end{aligned} \tag{7}$$

which is the average energy per bit of one pulse, at the receiver, divided by the spectral density of the noise. This is a commonly used definition that also agrees with the one in the textbook. The average energy of one pulse is here denoted by $E_{\rm s}$ and k is the number of bits per symbol ($k = 2$ for QPSK).

Since the up-conversion and the down-conversion basically cancel each other, all the blocks inside the dashed rectangle can be replaced by a *baseband equivalent channel*. This channel is sufficient for the purpose of simulating the system. The equivalent channel will now be derived in two steps by considering the transmitted signal and the noise separately. Computation of the SNR in terms of the equivalent system will also be dealt with.

2.3.1 Signal Part

In order to derive the baseband equivalent channel and compute the SNR it is sufficient to consider only a single transmitted pulse. Hence, the baseband signal after the PAM can be written as

$$s_{\rm bb}(t) = d(n)p(t) = (d_{\rm I} + jd_{\rm Q})p(t) \ , \tag{8}$$

with $d(n) = d_{\rm I} + jd_{\rm Q}$ and where $d_{\rm I}$ and $d_{\rm Q}$ represent the in-phase and quadrature-phase component of the transmitted symbol, respectively. The time index n has here been omitted to simplify the notation. A frequency domain illustration of the baseband signal, the passband signal and the corresponding received baseband equivalent signal (without noise) is given in Figure 4. Pay close attention

to how the spectrum of $s_{\mathrm{bb}}(t)$ is scaled and translated in frequency, resulting in the passband signal $s(t)$. From the spectrum of $r_{\mathrm{bb}}(t)$, it is clear that after the down-converter the original baseband signal $s_{\mathrm{bb}}(t)$ is retained.

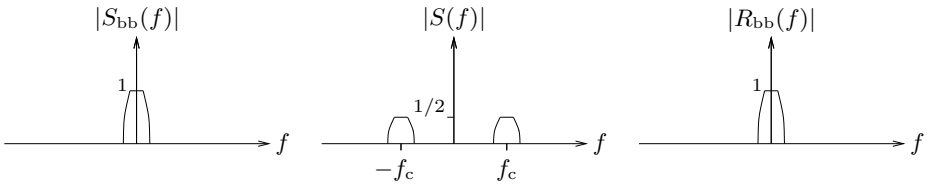


Figure 4: The Fourier transform of a pulse at the output of the PAM, the up-converter and the down-converter, respectively.

What remains is to compute the average energy of the passband signal in terms of the baseband pulse function $p(t)$. In view of (2), the passband signal can be written as

$$s(t) = d_{\mathrm{I}} p(t) \cos(2\pi f_{\mathrm{c}} t) - d_{\mathrm{Q}} p(t) \sin(2\pi f_{\mathrm{c}} t) \ . \tag{9}$$

The Fourier transform of the signal is then

$$S(f) = \frac{d_{\mathrm{I}}}{2}[P(f+f_{\mathrm{c}}) + P(f-f_{\mathrm{c}})] - j\frac{d_{\mathrm{Q}}}{2}[P(f+f_{\mathrm{c}}) - P(f-f_{\mathrm{c}})] \ . \tag{10}$$

By utilizing Parseval's theorem and that the signal constellation is normalized, the average energy of the passband signal is finally obtained as

$$\begin{aligned} E_{\mathrm{s}} &= \mathrm{E}\left[\|s(t)\|^2\right] = \mathrm{E}\left[\int |S(f)|^2 df\right] \\ &= \mathrm{E}\left[2\left(\frac{d_{\mathrm{I}}^2}{4}\int |P(f)|^2 df + \frac{d_{\mathrm{Q}}^2}{4}\int |P(f)|^2 df\right)\right] \\ &= \frac{\mathrm{E}[d_{\mathrm{I}}^2 + d_{\mathrm{Q}}^2]}{2}\int |P(f)|^2 df \\ &= \frac{1}{2}\int |P(f)|^2 df \\ &= \frac{1}{2}\|p(t)\|^2 \ . \end{aligned} \tag{11}$$

since we have assumed $E[d_{\mathrm{I}}^2 + d_{\mathrm{Q}}^2] = 1$. It is easy to verify this result in Figure 4 by comparing $S(f)$ with $S_{\mathrm{bb}}(f)$, while taking the scaling of the spectrum into account.

2.3.2 Noise Part

The baseband equivalent channel also needs to model the noise. For now, only the noise signal is assumed to be present in the received signal.

Care needs to be taken when representing additive white Gaussian noise (AWGN) in baseband. In order to obtain the baseband equivalent representation of the noise, the power spectral density of the noise after the down-converter

is sought. From the textbook, it is well-known that the power spectral density of the real-valued noise in the passband representation occupies both positive and negative frequencies, as illustrated in the left part of Figure 5. Below, some intuitive arguments of how to arrive at the power spectral density of the baseband equivalent noise are given.

The down-converter starts by multiplying $r(t)$ by $2e^{-j2\pi f_c t}$. Since the power spectral density of the passband noise is constant, the frequency shifting leaves the power spectral density unaffected, except that it is scaled by a factor of four (due to the doubling of the spectrum when going from passband to baseband equivalent and that the power spectral density is based on "the square of the Fourier transform"). Hence, a constant power spectral density of $2N_0$ is input to the low-pass filter. As a result, the power spectral density at the output of the down-converter is centered around zero with a height of $2N_0$ and a bandwidth $W_{\rm LP}$, equal to the bandwidth of the low-pass filter. This is illustrated in the right part of Figure 5.

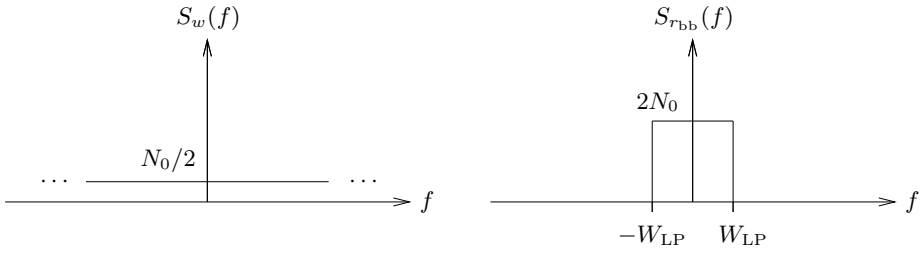


Figure 5: The power spectral density of the noise before and after down-conversion, respectively. The bandwidth of the low-pass filter is denoted by $W_{\rm LP}$.

From the above discussion, it is clear that the baseband equivalent noise may be represented by a complex Gaussian noise process with a constant power spectral density of $2N_0$. We do not need to bandlimit the noise to $-W_{\rm LP} < f < W_{\rm LP}$, since the output of the matched filter is the same, regardless if the noise is bandlimited or not (since the bandwidth of the low-pass filter is assumed to be at least as large as the bandwidth W of the matched filter).

A more strict derivation of how to obtain the the baseband power spectral density from the passband power spectral density can be found in the textbook. This gives that the baseband equivalent noise can be modeled as

$$w_{\rm bb}(t) = n_{\rm I}(t) + jn_{\rm Q}(t) \ , \tag{12}$$

where $n_{\rm I}(t)$ and $n_{\rm Q}(t)$ are two independent zero-mean white Gaussian noise processes with autocorrelation functions given by

$$r_{n_{\rm I}}(\tau) = r_{n_{\rm Q}}(\tau) = N_0\delta(\tau) \ . \tag{13}$$

Here, $\delta(\tau)$ is the Dirac delta function. Both quadrature components thus have a power spectral density equal to N_0. The previous intuitive results can now be verified by computing the autocorrelation function and taking the Fourier transform. Since the signals are complex valued, the autocorrelation function

of $w_{\rm bb}(t)$ is computed as

$$\begin{aligned} r_{w_{\rm bb}}(\tau) &= \mathrm{E}[w_{\rm bb}(t)w_{\rm bb}^*(t-\tau)] \\ &= \mathrm{E}[(n_{\rm I}(t)+jn_{\rm Q}(t))(n_{\rm I}(t-\tau)-jn_{\rm Q}(t-\tau))] \\ &= \mathrm{E}[n_{\rm I}(t)n_{\rm I}(t-\tau)] + \mathrm{E}[n_{\rm Q}(t)n_{\rm Q}(t-\tau)] \\ &\quad + j(\mathrm{E}[n_{\rm Q}(t)n_{\rm I}(t-\tau)] - \mathrm{E}[n_{\rm I}(t)n_{\rm Q}(t-\tau)]) \\ &= 2N_0\delta(\tau) \ , \end{aligned} \tag{14}$$

were the last inequality is due to (13) and the fact that the quadrature components are independent. Finally, Fourier transforming $r_{w_{\rm bb}}(\tau)$ shows that the power spectral density is constant with height $2N_0$. Thus, this confirms the intuitive result above.

The signal and noise only cases may now be combined to form the baseband equivalent communication system shown in Figure 6. As seen, the system has been considerably simplified due to the removal of both the up-converter and the down-converter. Furthermore, the noise is now complex valued with power spectral density $2N_0$.

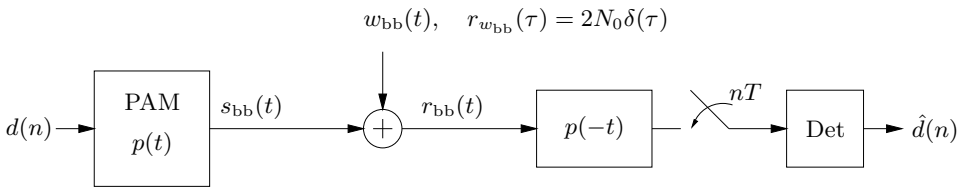


Figure 6: The baseband equivalent system.

As a final step, the SNR in terms of the parameters of the baseband equivalent system is computed. Taking the expression for the SNR in (7) and combining with (11) yields

$$\mathrm{SNR} = \frac{E_{\rm s}}{kN_0} = \frac{\|p(t)\|^2}{2kN_0} \ . \tag{15}$$

3 Simulating a Continuous-Time System

In this section, it is shown how the continuous-time baseband equivalent system in Figure 6 can be transformed into an equivalent discrete-time system. This is necessary in order to be able to simulate the system using, for example, Matlab, where only time discrete signals can be represented.

A simulation is often based on an *oversampled* system, i.e., the sampling rate is higher than the symbol rate. In general, a higher sampling rate will more accurately reflect the original system. However, this comes at the cost of a longer simulation time since more samples needs to be processed. It is common to use an oversampling rate that is a multiple of the symbol rate. The number of samples per symbols, here denoted by Q, is then an integer.

In order to arrive at the desired discrete time system, we will take the continuous-time baseband equivalent system, introduce an ideal anti-alias filter at the output of the matched filter and then oversample its output. This

is depicted in Figure 7, where also down-sampling a factor Q is performed in order to have a symbol sampled signal at the input of the detector. Hence, the oversampling is canceled by the down-sampling device. The oversampling factor Q is assumed to be chosen so large that the bandwidth of the matched filter is smaller than the bandwidth $Q/2T$ of the anti-alias filter. Consequently, the anti-alias filter does not change the signal output from the matched filter. The signal at the input of the detector is the same as for the continuous-time system. Thus, this oversampled system is equivalent to the original system.

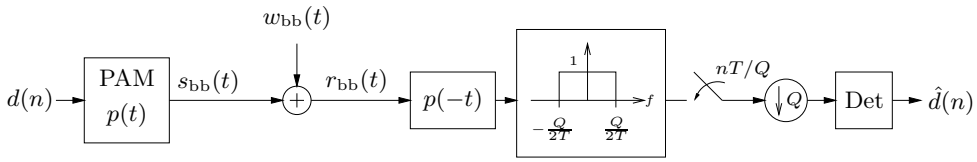


Figure 7: Oversampling the system.

Below, a series of equivalent systems are presented where the anti-alias filter and the sampling device, step by step, is moved to the left until a completely discrete-time system remains. As illustrated in Figure 8, the first step is to switch the order of the matched filter and the anti-alias filter and also use a discrete time matched filter $p(-n)$. With a slight abuse of notation, $x(n)$ denotes the discrete-time signal $x(nT/Q)$. This notation will be used extensively in the following. Scaling a factor T/Q is necessary in order to have the same signal level. However, for simulation purposes, this scaling operation can be omitted since both the signal and the noise is scaled by the same factor, thus leaving the SNR unchanged. The matched filter has a bandwidth smaller than the Nyquist frequency $Q/2T$ so this reordering operation does not affect the signal. A mathematical motivation for this is provided by considering the sampled output of the continuous-time convolution

$$y(t) = h(t) * x(t) \tag{16}$$

and approximating the integral with a summation to yield

$$\begin{aligned} y(nT_{\mathrm{s}}) = &\int_{-\infty}^{\infty} h(\tau)x(nT_{\mathrm{s}} - \tau)d\tau \\ \approx &T_{\mathrm{s}} \sum_{k=-\infty}^{\infty} h(kT_{\mathrm{s}})x((n-k)T_{\mathrm{s}}) \\ = &T_{\mathrm{s}} h(n) * x(n) \ . \end{aligned} \tag{17}$$

The relation holds exactly if both $h(t)$ and $x(t)$ have a bandwidth less than than the Nyquist frequency $1/2T_{\mathrm{s}}$. As a result, sampled continuous-time convolutions can be computed using discrete-time processing if both the input signal and filter have a bandwidth less than the Nyquist frequency (which is the case in our system) and if the output is scaled by the sample period T_{s}.

As the next step, it should be obvious that the anti-alias filter and the sampler can be moved in front of the summation without changing the signal at the detector. This transformation is shown in Figure 9. The anti-alias filter

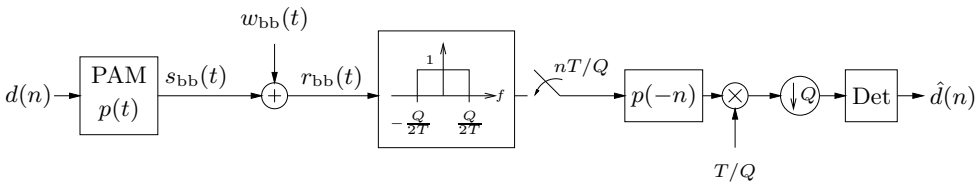


Figure 8: Moving the anti-alias filter and the sampling device in front of the matched filter.

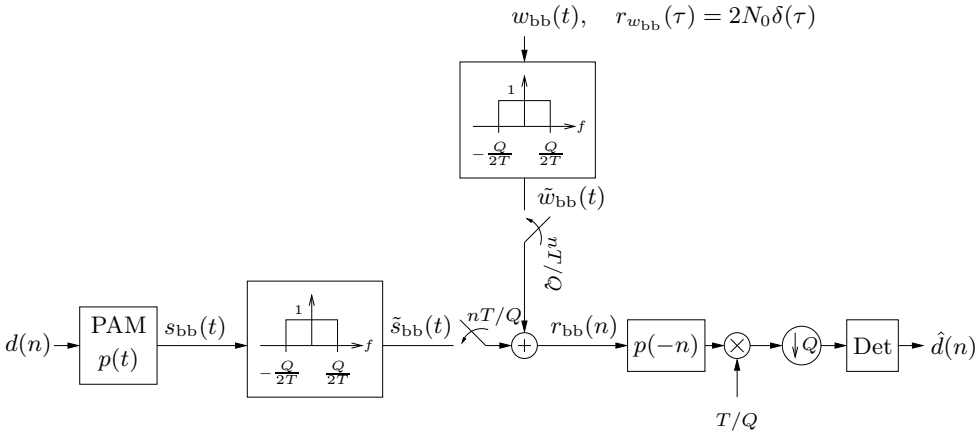


Figure 9: Sampling the transmitted signal and the noise separately.

after the PAM can be removed (since the bandwidth of $p(t)$ is smaller) which means that we can write the sampled transmitted signal as

$$\tilde{s}_{\rm bb}(nT/Q) = \sum_{k=-\infty}^{\infty} d(k)p(nT/Q - kT) = \sum_{k=-\infty}^{\infty} d(k)p((n-kQ)T/Q) \ . \quad (18)$$

Clearly, this summation can be implemented by up-sampling $d(n)$ by a factor of Q, followed by filtering the resulting sequence with $p(n)$.

For the noise, on the other hand, the autocorrelation function after the anti-alias filter is needed. A useful, well-known, relation between the power spectral density of the input and the output of a filter $h(t)$ is

$$y(t) = h(t) * x(t) \quad \Rightarrow \quad S_y(f) = |H(f)|^2 S_x(f) \ , \quad (19)$$

where $x(t)$ is the input and $y(t)$ the output of the filter. For the problem at hand, $x(t) = w_{\rm bb}(t)$, $h(t)$ is the anti-alias filter and $y(t) = \tilde{w}_{\rm bb}(t)$. The above relation combined with (14) shows that the power spectral density of $\tilde{w}_{\rm bb}(t)$ is

$$S_{\tilde{w}_{\rm bb}}(f) = \begin{cases} 2N_0, & |f| < Q/2T \\ 0, & \text{otherwise.} \end{cases} \quad (20)$$

To obtain the corresponding autocorrelation function, the inverse Fourier trans-

form is taken in order to arrive at

$$r_{\tilde{w}_{\rm bb}}(\tau) = \frac{2N_0Q}{T}\,\mathrm{sinc}\left(\frac{Q\tau}{T}\right) \ . \tag{21}$$

The autocorrelation for the sampled signal, $\tilde{w}_{\rm bb}(n)$, is obtained by sampling the expression in (21) using a sampling period T/Q,

$$r_{\tilde{w}_{\rm bb}}(k) = \frac{2N_0Q}{T}\,\mathrm{sinc}(k) = \frac{2N_0Q}{T}\delta(k) \ , \tag{22}$$

where $\delta(k)$ is the discrete-time delta function. As seen, the discrete-time noise is white with variance $2N_0Q/T$.

It is finally possible to summarize the above development in the equivalent discrete-time system shown in Figure 10, where $\tilde{s}_{\rm bb}(n)$ and $\tilde{w}_{\rm bb}(n)$ are written as $s_{\rm bb}(n)$ and $w_{\rm bb}(n)$, in order to simplify the notation, and the scaling after the matched filter is removed. This system is easily simulated using a computer. The noise variance added in the discrete-time model is obtained from (22), i.e.,

$$\sigma^2 = \frac{2N_0Q}{T} \ . \tag{23}$$

The quadrature components are independent white Gaussian noise processes with half this variance. Thus, the added noise can be written as

$$w_{\rm bb}(n) = w_{\rm bbI}(n) + jw_{\rm bbQ}(n) \ , \tag{24}$$

where

$$r_{w_{\rm bbI}}(k) = r_{w_{\rm bbQ}}(k) = \frac{\sigma^2}{2}\delta(k) \ . \tag{25}$$

From (15) and (23), the SNR is given by

$$\mathrm{SNR} = \frac{\|p(t)\|^2}{2kN_0} = \frac{Q\int_{-\infty}^{\infty}|p(t)|^2dt}{kT\sigma^2} = \frac{Q\sum_k|p(n)|^2T/Q}{kT\sigma^2} = \frac{\|p(n)\|^2}{k\sigma^2} \ , \tag{26}$$

where the third equality is due to the fact that the integral can be replaced by a summation, similarly to (17) but with T/Q instead of $T_{\rm s}$.

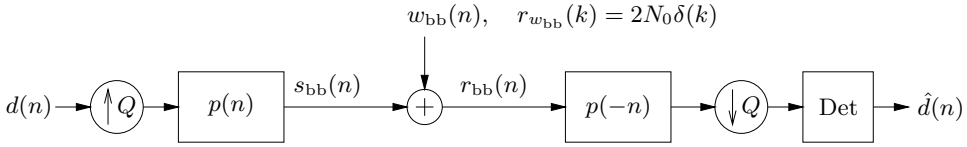


Figure 10: The equivalent discrete time baseband system.

4 System Description

The QPSK system considered in this report, illustrated in Figure 11, consists of a bit source, transmitter, channel, receiver, and a bit sink. The bit source

generates a stream of information bits to be transmitted by the transmitter. Typically, a random bit generator is employed as a bit source in simulations and this is the case herein as well. The transmitter converts the bits into QPSK symbols and applies optional pulse shaping and upconversion. The output from the transmitter is fed through a channel, which in its simplest form is an AWGN channel. The receiver block takes the output from the channel, estimates timing and phase offset, and demodulates the received QPSK symbols into information bits, which are fed to the bit sink. Typically, in a simulation environment, the bit sink simply counts the number of errors that occurred to gather statistics used for investigating the performance of the system.

In the following sections, the operation of the transmitter, channel and receiver are described in detail. Your task is to implement one or several transmitter and receiver algorithms, investigate their operation and performance and write a short technical report on your results.

Figure 11: The QPSK system considered.

4.1 Transmitter

The transmitter is illustrated in Figure 12 and consists of blocks for training sequence generation, QPSK modulation, pulse shaping, and carrier modulation. When simulating, usually only the baseband part is simulated, as previously discussed. However, in a real implementation, the passband part is essential and a thorough understanding of its properties is necessary in order to perform baseband simulations in a correct way.

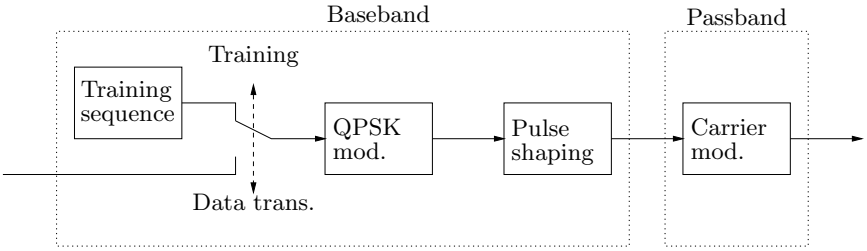


Figure 12: The QPSK transmitter.

4.1.1 Training Sequence

The training sequence generator generates a known data sequence which is transmitted prior to any data transmission. Its purpose is to provide the receiver

with a known sequence, which can be used for phase estimation and synchronization. The desired properties of a training sequence will be touched upon when discussing the synchronization algorithm. Basically, correlation function equal to the delta pulse is desired.

The training sequence is multiplexed with the data sequence before QPSK modulation. Herein, the multiplexing is done such that the whole training sequence is transmitted before the data sequence, but other schemes are of course possible as well. Putting the data sequence in the middle of the data, i.e. half the data bits, followed by the training sequence, followed by the other half of data bits, is a common scheme.

Training sequences carry no information and is therefore to be seen as "useless" overhead. A shorter training sequence is preferable from an overhead point of view, while a longer one usually results in better performance of the synchronization and phase estimation algorithms in the receiver.

4.1.2 QPSK Modulation

The bits are mapped onto corresponding QPSK symbols using Gray coding, as shown in Figure 13. Each QPSK symbol is represented by a complex number $d_{\mathrm{I}} + jd_{\mathrm{Q}}$, corresponding to the real-valued I and Q channels, respectively. However, as discussed in Section 2.1, it is common to use complex numbers for representing communication signals as this will simplify the analysis.

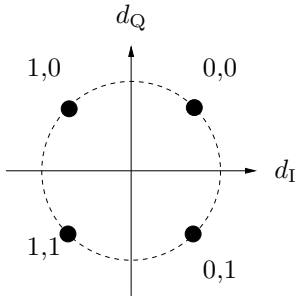


Figure 13: The mapping of bits $b(2n-1), b(2n)$ into QPSK symbol n.

4.1.3 Pulse Shaping

The resulting QPSK symbols are passed through an optional pulse shaping filter, where a suitable pulse shape can be applied if desired. Often, a rectangular pulse shape is used in simulations, although a (truncated) root raised cosine pulse is a common choice in a real system. The reason is the much improved spectral properties of a root raised cosine pulse compared to a rectangular one. The root raised cosine pulse shape is given by

$$p(t) = 4\beta \frac{\cos[(1+\beta)\pi t/T] + (\sin[(1-\beta)\pi t/T])/(4\beta t/T)}{\pi\sqrt{T}(1-16\beta^2 t^2/T^2)} \tag{27}$$

where β is called the roll-off factor. Both a rectangular pulse and a root raised cosine pulse (with $\beta = 0.22$) are shown in Figure 14.

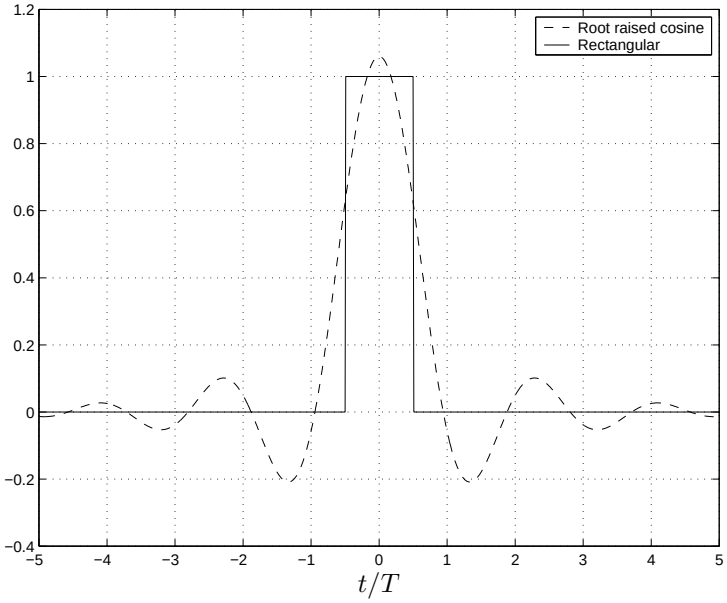


Figure 14: Two different pulse shapes.

4.1.4 Carrier Modulation

Subsequent to pulse shaping is carrier modulation, taking the complex valued pulse shaped QPSK symbols in the baseband, shifting them in frequency and outputting a real valued passband signal. The passband signal is simply generated by multiplying the complex valued baseband signal with a complex carrier of frequency f_c and taking the real part of the product. Note that the baseband signal is complex-valued, while the passband signal is real valued, as discussed in Section 2.1. Which carrier frequency to choose depends on the channel encountered, but the carrier frequency is almost always substantially higher than the baseband frequency determined by the symbol rate. When simulating communication system, it is common to perform the simulations in the baseband instead of in the passband. The reasons for this are several. One is the drastically increased simulation time required for passband simulations, due to the high sampling rate that is required. Another it that a baseband simulation can be used as a model for all different carrier frequencies, which is an advantage.

The carrier modulation block will not be simulated in Matlab, but is essential in a real system and hence it is necessary to understand its operation.

4.2 Channel

Several different channels are possible, the simplest one being the AWGN channel in Figure 15. This channel simply adds white Gaussian noise to the signal. Another common channel model is a multipath channel, shown for the two-path case in Figure 16. This channel causes intersymbol interference in the received signal as well, which will deteriorate the performance.

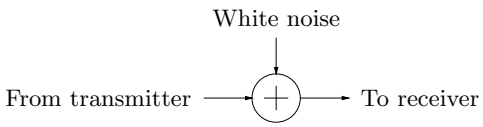


Figure 15: An AWGN channel.

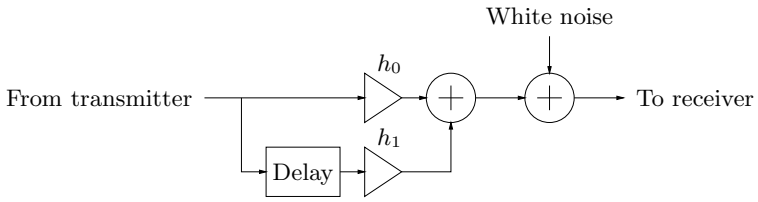


Figure 16: A simple two-path channel.

4.3 Receiver

The receiver is the most complex part in the system and is depicted in Figure 17. It consists of two main parts, a continuous-time front end, usually implemented with analog components, and a discrete-time baseband processing part, commonly implemented in a digital signal processor and/or ASICs. The front end consists of bandpass filtering (BP), down-conversion and sampling to form a discrete-time signal. The baseband part consists of matched filtering (MF), phase estimation, synchronization, sampling, phase correction, and decision. Only the baseband part is usually simulated, although the front-end is crucial for the performance and a thorough understanding of its properties is required.

4.3.1 Bandpass Filtering

The first block in the receiver is a bandpass filter with a center frequency equal to the carrier frequency f_c and a bandwidth matching the bandwidth of the transmitted signal (as given by the pulse shape used). The purpose of the bandpass filter is to remove out-of-band noise. Choosing the bandwidth of the bandpass filter requires some care. If the bandwidth is chosen too large, more noise energy than necessary will pass on to the subsequent stages. If it is too narrow, the desired signal will be distorted. The bandwidth of the transmitted signal can be defined in several different ways, which must be taken into account when choosing the filter bandwidth in the receiver. Furthermore, the phase characteristic of the bandpass filter must be such that the phase is not severely distorted as it carries the desired information.

The bandpass filter is part of the front end and will not be included in the simulation.

4.3.2 Down Conversion and Sampling

The down conversion block simply down-converts the received signal to the baseband, resulting in a complex-valued baseband signal. Down-conversion is a simple operation, just multiplying the input signal with the local oscillator

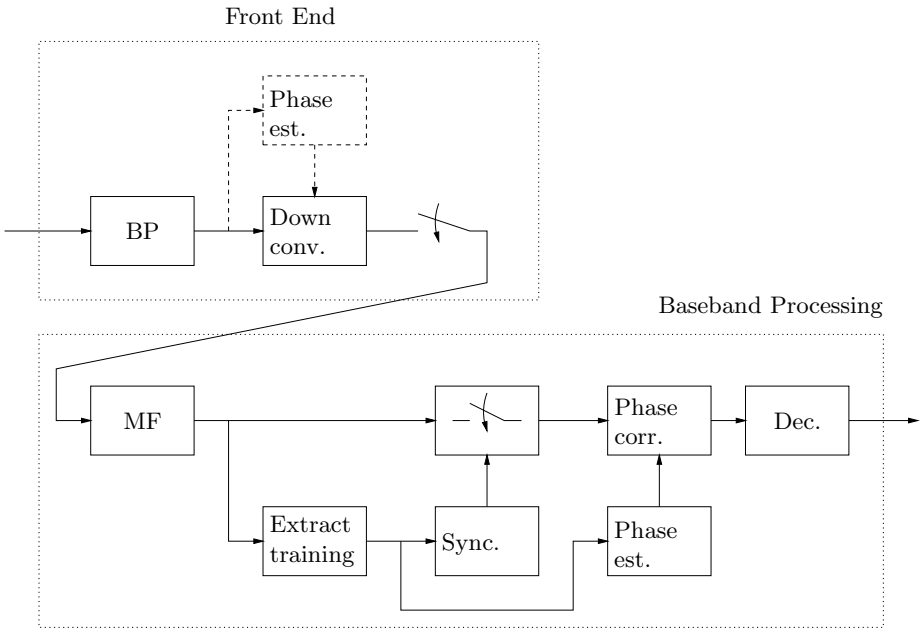


Figure 17: A possible receiver for the QPSK system.

signal, followed by low-pass filtering to remove terms of frequency $2f_c$ and fulfill the sampling theorem when sampling and converting into a discrete-time signal. The sampling rate is a multiple of the symbol rate. Its purpose is to obtain a discrete-time signal, which is more easily processed, for example using a digital signal processor. As shown in previous courses, a continuous-time signal can equivalently well be represented by a discrete-time one as long as the sampling theorem is fulfilled, which is ensured by choosing a sufficiently high sampling frequency (typically four or eight times the symbol rate). Note that this sampling process only converts the signal into the discrete domain. It is *not* related to the sampling of the matched filter.

One aspect of the local oscillator signal in the down-conversion block is how to set the initial phase. In Figure 17, a connection from the optional phase estimation block to the down-conversion block is shown with dashed lines and the phase estimate obtained from the phase estimator is used for the initial phase of the local oscillator. Another approach, which is more common in practice, is not to lock the phase of the local oscillator, but instead to do a phase compensation of the baseband signal. This can be done, for example, after the matched filtering, where a phase compensation simply is a rotation of the signal constellation. The latter approach is shown with solid lines in Figure 17.

Down-conversion will not be simulated as it is not part of the baseband system.

4.3.3 Matched Filtering

The matched filtering block contains a filter matched to the transmitted pulse shape. In the system described herein, the matched filter operates on a discrete-time signal. The two possibilities are equivalent (provided the sampling theorem discussed above is fulfilled, see Section 3), but from an implementation point of view, operating on the discrete-time signal is to prefer.

In case of a rectangular pulse shape, the matched filter is an integrate-and-dump filter. In Figure 18, the output signal from the matched filter (either the I or Q channel) is shown for the case of rectangular pulse shapes. The black dots represent the sampled signal in the receiver, assuming four samples per symbol. The optimal sampling instants are illustrated with small arrows. In the figure, the sampling of the matched filter happens to be at one of the samples of the discrete signal, but this is typically not the case. If the matched filter is to be sampled between two solid dots, interpolation can be used to find the value between two samples, or, simpler but with a loss in performance, the closest sample can be chosen.

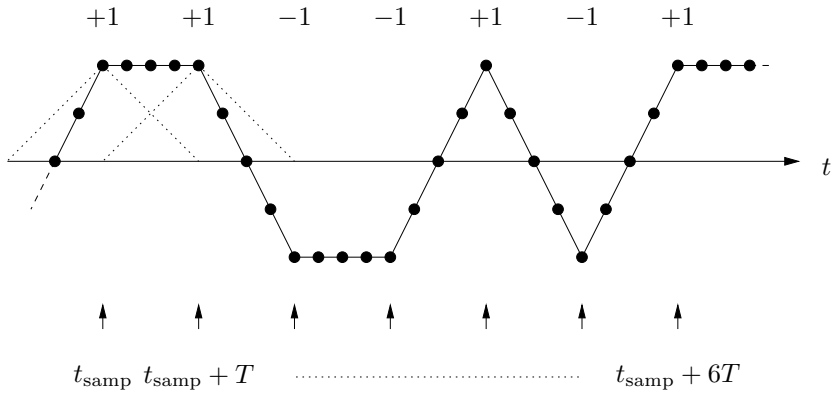


Figure 18: Output from the matched filter for successive signaling in absence of noise. The solid line is the resulting output signal from the matched filters and the dotted lines are the contributions from the two first bits (the remaining bits each have a similar contribution, but this is not shown). The small arrows illustrate the preferred sampling instants.

4.3.4 Synchronization

The synchronization algorithm is crucial for the operation of the system. Its task is to find the best sampling time for the sampling device. Ideally, the matched filter should be sampled such that the SNR for the decision variable is maximized. For a rectangular pulse shape, the best sampling time t_{samp} is at the peak of the triangles coming out from the matched filter, illustrated with small arrows in Figure 18.

The synchronization algorithm used herein is based on the complex training sequence. During the training sequence, it is known to the receiver what the transmitter is transmitting. Hence, one possible way of recovering the symbol timing is to cross-correlate the complex-valued samples after the matched filter

with a locally generated time-shifted replica of the training sequence. By trying different time-shifts in steps of T/Q, where Q is the number of samples per symbol, the symbol timing can be found with a resolution of T/Q. Put into mathematical terms, if $\{c(n)\}_{n=0}^{L-1}$ is the locally generated symbol-spaced replica of the QPSK training sequence of length L, $[t_{\text{start}}, t_{\text{end}}]$ represents the search window and $r(n)$ denotes the output from the matched filter, the timing can be found as

$$t_{\text{samp}} = \arg \max_{t_{\text{samp}} \in [t_{\text{start}}, t_{\text{end}}]} \left| \sum_{k=0}^{L-1} r(kQ + t_{\text{samp}})^* c(k) \right| . \tag{28}$$

An example of what the correlation might look like is shown in Figure 19.

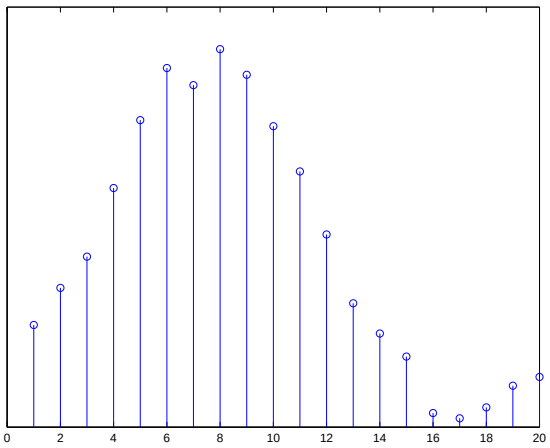


Figure 19: Example of cross-correlating the received sequence with the training sequence in order to find the timing. In this example, the delay was estimated to 8 samples (corresponding to the maximum) and, hence, the matched filter should be sampled at $8, 8+Q, 8+2Q, \ldots$ in order to recover the QPSK symbols.

The correlation properties of the training sequence are important as they affects the estimation accuracy. Ideally, the autocorrelation function for the training sequence should equal a delta pulse, i.e., zero correlation everywhere except at lag zero. Therefore, the training sequence should be carefully designed. However, in the lab it is simply chosen as a random sequence, which usually works fairly well if the length is large enough, but for a shorter length the correlation properties of a random sequence can be quite bad.

Note that the synchronization strategy outlined above is just one possibility and several other schemes are plausible as well.

4.3.5 Sampling

The output from the matched filter is down-sampled with a sampling rate of $1/Q$, i.e. every Q^{th} symbol in the output sequence is kept. The position for the samples (illustrated by arrows in Figure 18) is controlled by the synchronization device previously described.

4.3.6 Phase Estimation

The phase estimator estimates the phase of the transmitted signal, which is necessary to know in order to demodulate the signal (unless differential detection is used). Phase estimation, especially in the low SNR region, is a hard problem and several different techniques are available. In principle, any estimation algorithm described in the signal processing literature could be used. The algorithm described below uses the training sequence and is fairly easy to implement, but there are several other possibilities as well.

Using a complex baseband representation, the received symbols after sampling the matched filter is a sequence of the form

$$\cdots, e^{j(\Phi_{n-1}+\varphi)}, e^{j(\Phi_n+\varphi)}, e^{j(\Phi_{n+1}+\varphi)}, e^{j(\Phi_{n+2}+\varphi)}, \cdots \quad , \tag{29}$$

where $\Phi_n \in \{\pm\pi/4, \pm 3\pi/4\}$ is the information bearing phase of the n^{th} symbol and φ is the unknown phase offset caused by the channel (and various filters in the receiver). If Φ_n is known, which is the case during the training sequence, the receiver can easily remove the influence from the information in each received symbol by element-wise multiplication with the complex conjugate of a QPSK modulated training sequence replica, generated by the receiver. The value of φ can then easily be obtained by averaging over the sequence. In other words, if $\{\tilde{r}(n)\}_{n=0}^{L-1}$ denotes the L received QPSK symbols (i.e. received signal after downsampling) during the training sequence and $\{c(n)\}_{n=0}^{L-1}$ is the local replica of the complex training sequence, an estimate of the unknown phase offset can be obtained as

$$\hat{\varphi} = \frac{1}{L}\sum_{k=0}^{L-1} \arg\left(\tilde{r}(k)c(k)^*\right) \tag{30}$$

The longer the training sequence, the better the phase estimate as the influence from noise decreases. A long training sequence, on the other hand, reduces the amount of payload that can be transmitted during a given time.

Another approach, which avoids the phase estimation problem, is to use differential detection. In this case, the transmitted data is differentially encoded such that the information lies in the *relative change* of the phase, not the absolute value. This simplifies the receiver, but at the cost of decreased performance.

4.3.7 Decision

The decision device is a threshold device comparing the I and Q channels, respectively, with the threshold zero. If the decision variable is larger than zero, a logical "0" is decided and if it is less than zero a logical "1" is decided.

5 Simulation Chain

A skeleton to a simulation chain is provided and can be found as a zip file at the course web-page. Download the file and unpack it using, e.g., unzip. In addition to the main simulation program, `simulation.m`, most of the routines necessary for a working transmitter and channel are provided, together with some of the receiver functions. The missing functions have to be written by yourself. Of

course you are free to change any functionality in any of the modules in order to study different aspects of the QPSK system.

Documentation in the form of Matlab help text (type `helpdesk` or type `help` *function_name*) is available for all functions in the simulation chain. For the missing functions, only the help text is provided and the rest has to be written. After studying this manual, the other Matlab functions and the help text, it should be possible to write the missing code without too much effort.

Estimating the bit-error probability or bit-error rate (BER) at the receiver is easily done by counting the number of errors that has occurred and dividing with the total number of bits transmitted. Of course the estimate improves as the number of bits simulated increases, but as a rule of thumb at least 100, and preferably more, errors should occur for each value of E_b/N_0 in order to get a reasonable accuracy.

5.1 The Program `simulation.m`

```
% Skeleton code for simulation chain

% History:
%   2000-06-28  written /Stefan Parkvall
%   2001-10-22  modified /George Jongren

clear

% Initialization
EbN0_db = 0:10;                         % Eb/N0 values to simulate (in dB)
nr_bits_per_symbol = 2;                 % Corresponds to k in the report
nr_guard_bits = 10;                     % Size of guard sequence (in nr bits)
                                        % Guard bits are appended to transmitted bits so
                                        % that the transients in the beginning and end
                                        % of received sequence do not affect the samples
                                        % which contain the training and data symbols.
nr_data_bits = 1000;                    % Size of each data sequence (in nr bits)
nr_training_bits = 100;                 % Size of training sequence (in nr bits)
nr_blocks = 50;                         % The number of blocks to simulate
Q = 8;                                  % Number of samples per symbol in baseband

% Define the pulse-shape used in the transmitter.
% Pick one of the pulse shapes below or experiemnt
% with a pulse of your own.
pulse_shape = ones(1, Q);
%pulse_shape = root_raised_cosine(Q);

% Matched filter impulse response.
mf_pulse_shape = fliplr(pulse_shape);


% Loop over different values of Eb/No.
nr_errors = zeros(1, length(EbN0_db));   % Error counter
for snr_point = 1:length(EbN0_db)

  % Loop over several blocks to get sufficient statistics.
  for blk = 1:nr_blocks

    %%%
    %%% Transmitter
    %%%

    % Generate training sequence.
    b_train = training_sequence(nr_training_bits);

    % Generate random source data {0, 1}.
    b_data = random_data(nr_data_bits);

    % Generate guard sequence.
    b_guard = random_data(nr_guard_bits);
```

```
    % Multiplex training and data into one sequence.
    b = [b_guard b_train b_data b_guard];

    % Map bits into complex-valued QPSK symbols.
    d = qpsk(b);

    % Upsample the signal, apply pulse shaping.
    tx = upfirdn(d, pulse_shape, Q, 1);

    %%%
    %%% AWGN Channel
    %%%

    % Compute variance of complex noise according to report.
    sigma_sqr = norm(pulse_shape)^2 / nr_bits_per_symbol / 10^(EbN0_db(snr_point)/10);

    % Create noise vector.
    n = sqrt(sigma_sqr/2)*(randn(size(tx))+j*randn(size(tx)));

    % Received signal.
    rx = tx + n;

    %%%
    %%% Receiver
    %%%

    % Matched filtering.
    mf=conv(mf_pulse_shape,rx);

    % Synchronization. The position and size of the search window
    % is here set arbitrarily. Note that you might need to change these
    % parameters. Use sensible values (hint: plot the correlation
    % function used for syncing)!
    t_start=1+Q*nr_guard_bits/2;
    t_end=t_start+50;
    t_samp = sync(mf, b_train, Q, t_start, t_end);

    % Down sampling. t_samp is the first sample, the remaining samples are all
    % separated by a factor of Q. Only training+data samples are kept.
    r = mf(t_samp:Q:t_samp+Q*(nr_training_bits+nr_data_bits)/2-1);

    % Phase estimation and correction.
    phihat = phase_estimation(r, b_train);
    r = r * exp(-j*phihat);

    % Make decisions. Note that dhat will include training sequence bits
    % as well.
    bhat = detect(r);

    % Count errors. Note that only the data bits and not the training bits
    % are included in the comparison. The last data bits are missing as well
    % since the whole impulse response due to the last symbol is not
    % included in the simulation program above.
    temp=bhat(1+nr_training_bits:nr_training_bits+nr_data_bits) ~= b_data;
    nr_errors(snr_point) = nr_errors(snr_point) + sum(temp);

    % Next block.
  end

  % Next Eb/No value.
end

% Compute the BER.
BER = nr_errors / nr_data_bits / nr_blocks;
```

5.2 Provided functions

The Matlab functions below are provided in the zip file, together with the main program `simulation.m`. No documentation is provided in this report. Instead, the reader is referred to the m-files themselves and the help function in Matlab.

- `simulation.m`
  The main simulation program, listed in Section 5.1.
- `random_data.m`
  A function for generating random data bits to be transmitted.
- `training_sequence.m`
  A function for generating the training sequence. Currently, a random sequence is used, but there are better alternatives.
- `root_raised_cosine.m`
  Returns the root raised cosine pulse shape.

5.3 Missing Functions

The Matlab functions listed below are not provided, but have to be written. Note that in order to investigate the properties of the QPSK system, some additional functions might be necessary, e.g., plotting the BER, plotting the eye diagram, etc. For a detailed description of what the functions are supposed to do, please see the m-files themselves and the help text included.

- `d = qpsk(b)`
  A QPSK modulator, mapping pairs of bits $\{0, 1\}$ into complex-valued symbols.
- `t_samp = sync(mf, b_train, Q, t_start, t_end)`
  Determines when to sample the matched filter outputs.
- `phihat = phase_estimation(r, b_train)`
  Phase estimator operating on the received baseband signal.
- `bhat = detect(r)`
  Determines the received bits given a received sequence of (phase-corrected) QPSK symbols. Gray coding of the individual bits is assumed.

6 Technical Report

The result of your work is to be reported in a technical report, with any plots and/or tables you find necessary. A length of four or five pages should suffice, and the report is not to exceed six pages (excluding the cover page and the enclosed Matlab code; see below). The text should of course be clearly structured and easy to read. A good structure of the presentation, adhering to what is common for technical and scientific publications, is therefore necessary. Your results should also be possible to reproduce with reasonable effort based on your report. When grading the report, the technical content, as well as presentation and language qualities, are considered. At most two authors are allowed. Either Swedish or English is allowed as the language of choice. All Matlab code that has been written as part of the assignment must be enclosed as a separate enclosure (this is not considered to be part of the max-six-pages main report).

The template provided at the end of this document will be used to grade the report. Use this template as cover page when you hand in the report. The only possible grades are *pass* or *fail*.

6.1 Report Outline

A suggested outline of the report is:

- Cover Page
- Abstract
  A short summary of your report.
- Background and Problem Formulation
  What is the background to the problem? Which questions are you trying to answer?
- Methodology
  How are the questions answered? Simulations or analytical analysis? Which scenarios are relevant to investigate?
- Results
  The results from the analysis and simulations. This should not only be a collection of plots, but *should* also contain explanations of your results. Why does the system behave in a certain way? Are there any improvements that can be made to the algorithms? Why do not the simulation results agree with theory?
- Conclusions
  Which conclusions can be drawn? Suggestions for future research?
- Appendix
  Not all reports have an appendix, but lengthy derivations, excessive plots, etc that you want to include without cluttering the main presentation is preferably placed in an appendix.
- References
  References to other publications whose results you are using in your presentation.
- Separate enclosure with the Matlab code you have written.

6.2 Problem Areas

Except for the first item in the list below, there is no detailed list of questions you must answer. Instead, some suggestions, in no particular order, of issues that deserves a closer study are listed below. As a measure of "how much work you have to do to pass," try to address *at least* five of the issues listed, and, in addition, any other interesting observations you have made during your work (this, in addition to the six-page-limit, is the only specification of the expected extent of the work).

- Investigate the BER performance as a function of E_b/N_0 with perfect phase estimation and synchronization. Derive the exact expression for the corresponding BER and compare the expression you have derived with the estimated BER:s from your simulation. Does the simulations results agree with the theoretical results? If not, why? **Note: This issue is compulsory!**

- With realistic (non-ideal) phase estimation and synchronization, the performance will of course deteriorate. Investigate the performance of synchronization and phase estimation algorithms for various values of E_b/N_0, starting with the noise free case (E_b/N_0 very large). Which of the two estimates, phase and timing, is more sensitive to noise? Why? How can it be improved?

- Study the signal constellation in the receiver for various values of E_b/N_0. What happens when the noise level increases? What are the implications from an error in the phase estimate?

- Plot the power density spectrum of the transmitted signal (e.g., with the matlab function `psd`) with different pulse shapes. How is it affected if a root raised cosine pulse is chosen instead of a rectangular pulse shape?

- Write a small function that plots the eye diagram and study it for both the AWGN channel and for the simple two-path ISI channel. How is the eye diagram affected? What implications does the ISI have on BER, phase estimation accuracy and synchronization accuracy?

- Investigate the performance of phase estimation and synchronization for different lengths of the training sequence. What is a reasonable length for an E_b/N_0 in the range of 0 to 20 dB? Which overhead, i.e., the number of "useless" training bits relative to the total number of transmitted bits, does this correspond to?

- Introduce a time-varying phase shift in the channel (this is an effect that is commonly present in real-life radio channels) and investigate the performance for different amounts of phase shifts. A phase increase up to $\pi/4$ during the transmission of one data block can be a good choice. Where during the data block will most errors occur? Can differential demodulation alleviate this effect?

- Implement differential modulation/demodulation, i.e., a demodulator that does not require any phase estimate. Compare the performance with the demodulator using an explicit phase estimate for different signal-to-noise ratios.

- Try better training sequences than a random one. Will this help performance for short training sequences? If so, with how much?

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: .....

Student 2: .....

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

PASSED

Signature: