Guide Book – Project in Wireless Communication, Part I

Guoda Tian, Dept. of Electrical and Information Technology, Lund University

September 20, 2024

1 Task 1

The final goal of task 1 is to simulate a passband communication system that consists of a transmitter, the propagation channel, and a receiver. The block diagrams of this communication system are presented in both the lecture slides and in this document. To finish the task, you should understand the function of each block as a first step. Afterward, every single block should be properly implemented and connected. Finally, the bit error rate (BER) curve for different signal-to-noise ratios (SNRs) should be plotted. This is expected to follow the theoretical bit error rate expression. The code can be implemented in either Matlab or other high-level programming languages such as Python or C/C++.

1.1 Block Diagram of the System

Block diagrams of the transmitter and receiver are presented in Fig. 1 and Fig. 2, respectively.

1.1.1 Transmitter and Channel part

In the transmitter part, the block diagram is divided into seven subsections. Fig. 1 illustrates the relationship between these subsections.

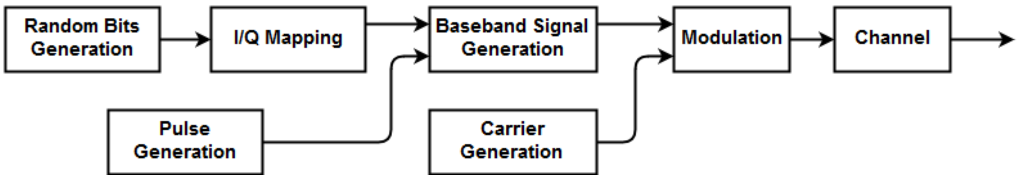

Figure 1: Transmitter block diagram.

1.1.2 Receiver Part

The block diagram of the receiver is presented in Fig. 2. As shown, the received signal and/or noise should be normalized with respect to a given SNR. After normalization, the signal should be down-converted, followed by matched filters. Afterwards, it is important to correctly sample the output signal according to the requirement of the matched filter. In the final step, the received signal should be demodulated and the bit error rate evaluated.

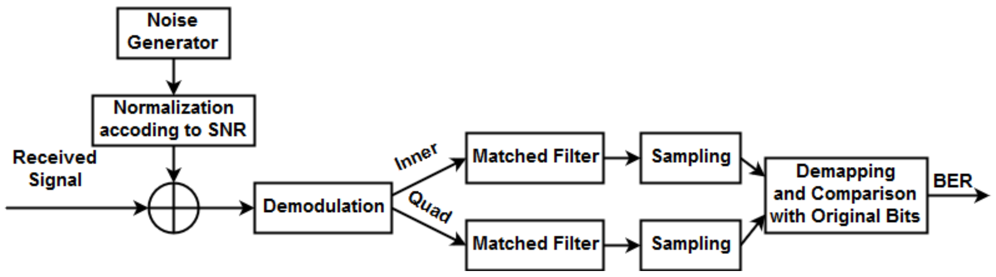

Figure 2: The receiver structure of task 1.

1.2 Detailed Explanation of the Transmitter and Channel Blocks

1.2.1 Random Bits Generation

A sufficient number of binary random bits should be generated for the sake of a smooth BER curve. Due to the requirement that the plot should visualize the error probabilities down to 10^{-4}, it is necessary to generate more than 10^6 bits. There are many ways to generate these random bits; please find a suitable solution yourself.

1.2.2 I/Q Mapping

Since we simulate a QPSK system, every two bits should be considered as a group and form one symbol (represented by a complex number). Then, the symbol should be mapped to the constellation diagram; see the digital communication book Fig. 5.2.

1.2.3 Pulse Generation

In our project, we select a half sine wave as our pulse shape, i.e.

$$p(t) = A\sin(\omega_p t), \omega_p t \in [0, \pi] \tag{1}$$

where $\omega_p = 2\pi f_p$. The duration of the symbol is set to $T_p = 2.3ms$. Based on this symbol time, it is your job to find the parameters ω_p and f_p.

After the calculation of the parameter f_p, you should program this base band pulse. Recall that the sampling frequency is given as F_s, the discrete time signal model for the pulse train can be written as:

$$p(n) = A * \sin(\omega_p n T_s), \tag{2}$$

where $T_s = \frac{1}{F_s}$. In addition, note that the last point of each pulse should be omitted, as indicated in the lecture notes.

1.2.4 Baseband Signal Generation

The next step is to generate the baseband signal. The lecture slide shows one feasible way to generate this signal by convolution. In addition, you may get help from the Kronecker product, which has been presented in the course *Multiple Antenna Systems.*

1.2.5 Carrier Generation and upconversion

To perform the upconversion, a carrier signal must be created (see slides). Observe that the carrier should also be discretized.

1.2.6 Propagation Channel

As shown in the project manual, the propagation channel is a two-tap channel. You may simulate the channel by convolution.
Questions:

- What is the coherence bandwidth of this channel?
- What is the coherence time of this channel?
- Is a narrow-band assumption satisfied?

1.3 Detailed Explaination of the Receiver Part

1.3.1 Noise Generator

Here you may refer to the Matlab function *randn*. Read about this built-in function yourself.

1.3.2 Normalization

To simulate a BER curve correctly, it is very important to normalize the signal and/or the noise, so that the SNR is exactly as expected. Theoretically, for an AWGN channel, the bit error probability can be expressed as:

$$P_b = \mathbf{Q}\left(\sqrt{d_{0,1}^2 \frac{E_b}{N_o}}\right) \tag{3}$$

In equation (3), $d_{0,1}^2$ is a constant value and the ratio $\frac{E_b}{N_o}$ is defined as the SNR. You should carefully calculate E_b based on your own code and add the corresponding noise level.
Hint: You may simplify the process by normalizing your signal in the pulse generation step.

1.3.3 Demodulation (Downconversion), Matched Filter and Sampling

Like the upconversion process, it is necessary to generate the same carrier (the same frequency) for the demodulation part. The matched filter can be implemented using the convolution operation or the Matlab function *filter*. Additionally, to fully exploit the power of the matched filter, the received signal should be sampled at the correct point, as shown in the lecture slides.

1.3.4 Demapping and Bit Comparison

As a final step, the original signal should be recovered and the bit error rate estimated. You should compare the simulation results with the theory. Keep in mind that the theoretical values are based on a system using QPSK with Gray code, in an AWGN channel; what does this mean for your comparison?

1.4 Example BER curve

One example of a BER plot is illustrated in Figure 3. If all the sub-blocks are implemented correctly, the gap between the theoretical and simulation curves is relatively small.

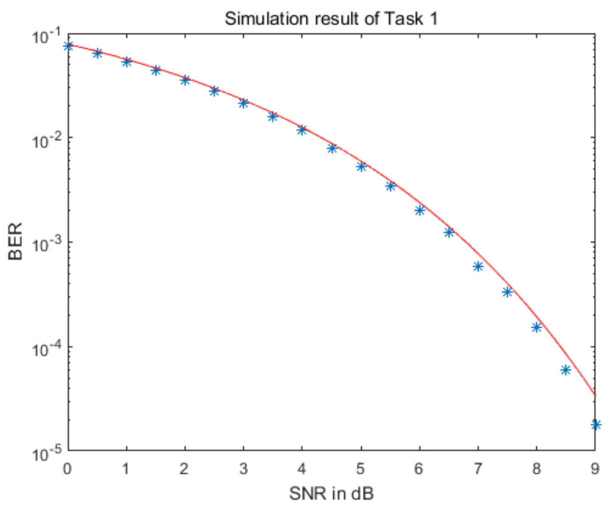

Figure 3: Example result of task 1.

2 Task 2

2.1 Introduction

Task 2 mainly deals with time synchronization for a simple non-multipath channel. As indicated in the lectures, the main goal for the time synchronization is to find the starting point of the received signal with the aid of pilots. The system model of the transmitter and channel is given in Fig. 4, while the receiver model is given in Fig. 5. The received signals can be downloaded from the course website. Therefore, in this task, you only need to implement the receiver to decode these signals.

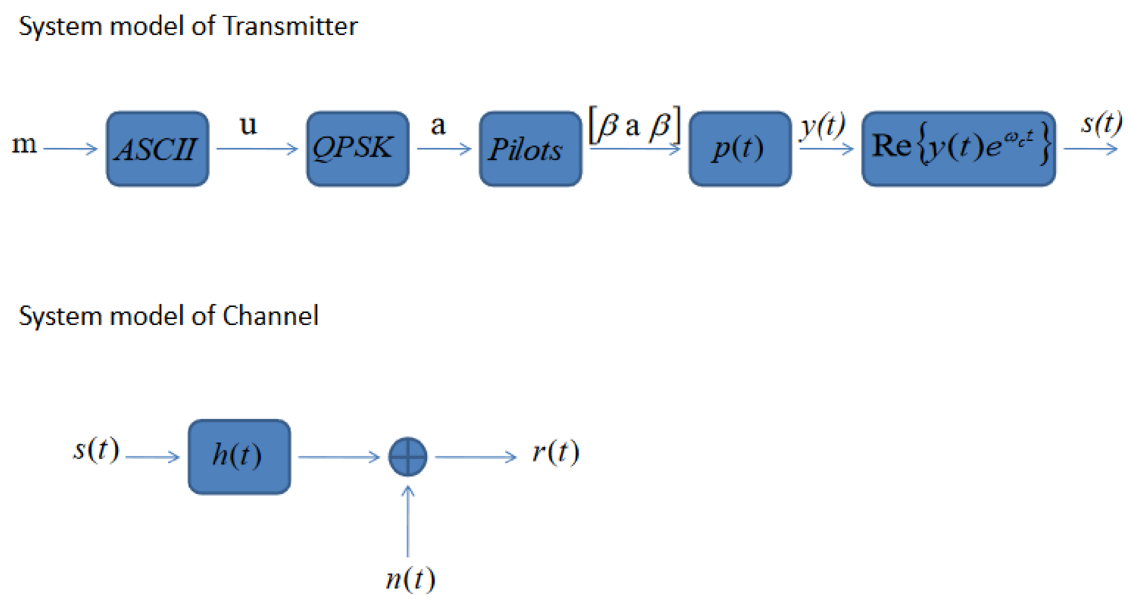

Figure 4: The transmitter and channel model.

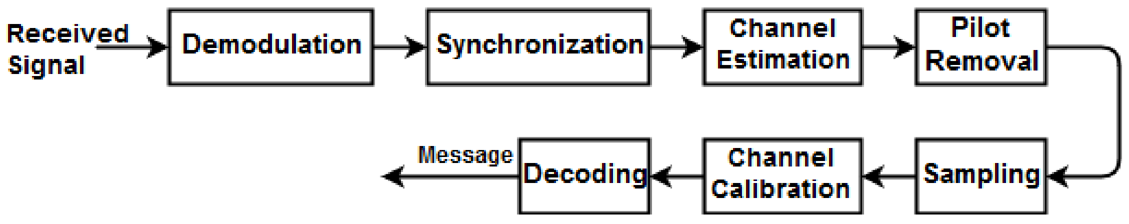

Figure 5: The receiver model.

2.2 Detailed Explanation of Each Block

2.2.1 Downconversion

The downconversion process is similar to the one in the first task; keep in mind that the *I-component* and the *Q-component* should be extracted. The following procedures work with a complex signal.

2.2.2 Synchronization

The main goal for the synchronization process is to find the start and end point of the signal within a long received sequence of samples. The method for synchronization has already been illustrated in the lecture notes.

2.2.3 Channel Estimation

An additional function of the pilot signal, in addition to synchronization, is to allow the estimation of the propagation channel. Hints provided below may be helpful:

- The propagation channel can be represented by a complex number.
- It is important to sample the pilot signal in a correct way. We recommend that you read the lecture notes carefully to get a solid understanding of the background knowledge.

2.2.4 Pilot Removal

The mission of the pilot signal has been fulfilled (in this task) after time synchronization and channel estimation are performed; therefore, it can be removed from the received sequence.

2.2.5 Sampling and Channel Equalization

To achieve the maximum gain of the matched filter, the received signal should be correctly sampled, similar to the first task. In addition, prior to the decoding step, the amplitude scaling and phase rotation of the channel should be compensated for (channel equalization) so that the transmit signal can be decoded.

2.2.6 Decoding

After channel equalization is performed, the original signal can be decoded. This is similar to task I. The code below may be helpful for the ASCII conversion; it is allowed to directly include this in your program.

```
wh=2.^[6:-1:0];
m=char(TransmittedBits(1:7)*wh');
for l=2:floor(length(TransmittedBits)/7),
m=[m char(TransmittedBits(7*(l-1)+1:7*l)*wh')];
end
m
```

3 Task 3

3.1 Introduction

The purpose of task 3 is to program a receiver for an OFDM based communication system. OFDM is a very popular transmission technology used by many standards such as LTE, 5G NR, and WiFi. Compared with task II, the propagation channel in task 3 is a frequency-selective fading channel, therefore, it is necessary to run a more complicated synchronization algorithm and possibly also a more advanced channel estimation and equalization process. The block diagrams of the transmitter and receiver are presented in Fig. 6 and Fig. 7, respectively. Similarly to the second task, the received signal can be downloaded from the course homepage, thus you only need to program the receiver part to decode this signal.

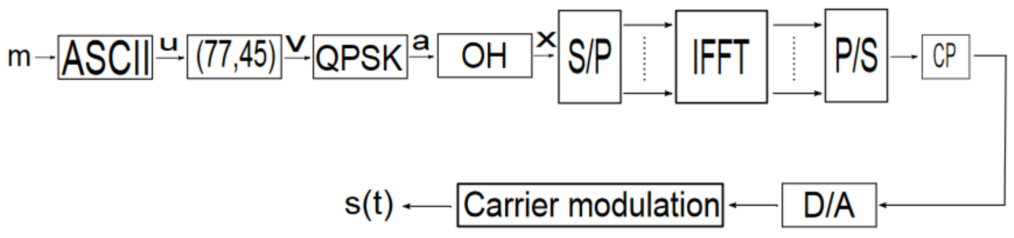

Figure 6: Block diagram of the OFDM transmitter.

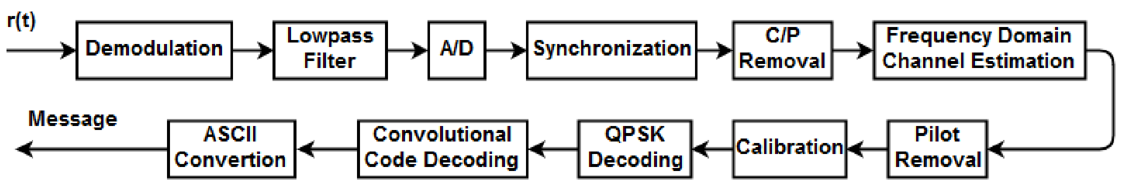

Figure 7: Block diagram of the OFDM receiver.

3.2 Detailed explanation of each block

3.2.1 Demodulation (Downconversion) and Lowpass Filtering

When it comes to down-conversion, the carrier frequency in this task is 10 kHz. Note that this carrier frequency should be changed later in task 4 to be suitable for real speakers and microphones.

Regarding the low-pass filter, you may refer to the Matlab code below as a reference, which is a digital filter with cutoff frequency 0.05. However, it is recommended to go back to the digital signal processing course to recap the basic concepts of filter design if you are unfamiliar with them.

```
[B,A]=butter(8,0.05);
FilteredSignal= filter(B,A,Signal);
```

Questions

- Plot the amplitude frequency response of this low-pass filter.

3.2.2 A/D conversion and Downsampling

The A/D conversion has already been done for you, therefore you just need to downsample the signal. We perform interpolation between two consecutive samples on the transmitter side. However, it is your job to figure out the correct upsampling factor in order to correctly downsample the signal.

Hint:

- Read your OFDM lecture note carefully.
- Please figure out the relationship between sampling frequency, number of subcarriers and OFDM symbol time.

3.2.3 Synchronization and Removal of the Cyclic Prefix

Due to the nature of the frequency-selective fading channel, it is challenging to use a simple matched filter for the synchronization task. The method for synchronization is clearly presented in the fifth lecture slides; however, it is necessary to carefully investigate the mathematical formulas, especially when it comes to the implementation of the integral operation in discrete time.

After figuring out the starting point of the signal, the cyclic prefix can be removed. As stated in the lecture notes, it is allowed to estimate the starting point somewhat earlier but not later than the ideal starting point, please explain the reason.

Hint:

- Matlab indices start from 1 instead of 0. Python and C/C++ indexes start from 0.
- Pilot symbols also need a cyclic prefix to protect them from ISI, do not forget it!

3.2.4 Channel Estimation, Pilot Removal and Calibration

Similar to the previous task, an important feature of the pilot is to enable a reliable estimation of the channel. Note that in this task, the estimator works in the frequency domain due to the OFDM operation. As illustrated in the project description, half of the channels have pilots. Therefore, interpolation is needed. After the estimation process has been completed, the pilot symbols can be removed from further processing. The channel effect should be compensated for before decoding the payload data.

3.2.5 Decoding and ASCII Conversion

After equalization, QPSK decoding should be implemented, followed by decoding the convolutional code. As indicated in the lecture slides, you may use the built-in function 'vitdec' for help, but it is important to have a solid understanding of this built-in function when you present and implement your work.

Similarly to the previous task, you may use the same code for ASCII conversion. However, unlike the second task, the message length should be extracted first.