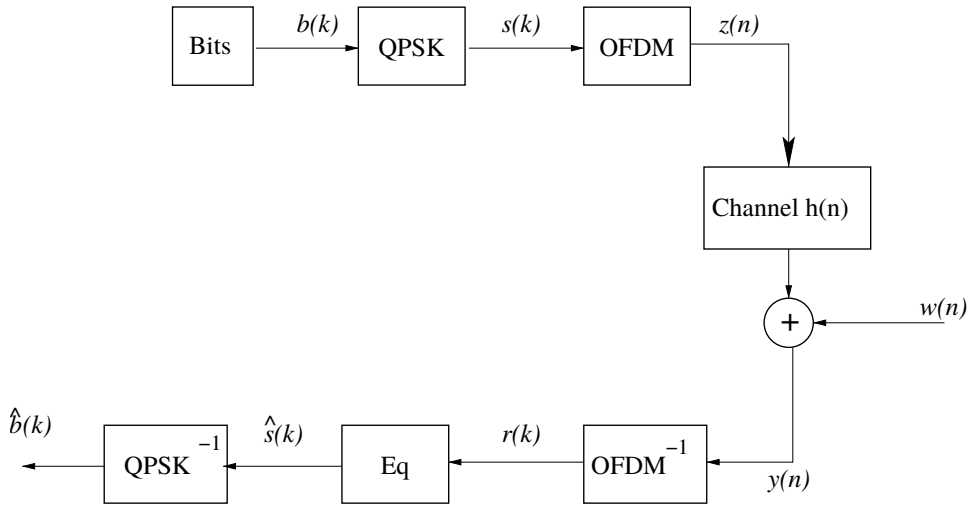
Applied Signal Processing

Project 1 - Acoustic Communication System

T. McKelvey, I. Andersson, A. Ozcelikkale, J. Lock

November 2018, version 1.1



Contents

1 Administrative details **2**
1.1 Purpose . . . 2
1.2 Project examination . . . 3
1.3 Deadlines and oral examination . . . 3

2 Project 1 overview **3**
2.1 Introduction . . . 3
2.2 Tasks in Part A and Part B . . . 4

3 Part A: Baseband Communication **5**
3.1 QPSK coding . . . 5
3.2 The physical channel . . . 6
3.2.1 Channel model . . . 6
3.3 Time domain equalization . . . 7
3.4 OFDM based communication . . . 7
3.5 Cyclic prefix . . . 9
3.6 Equalization with known H(k) . . . 9
3.7 Equalization with unknown H(k) . . . 10
3.8 Tasks Part A . . . 11

4 Part B: Interpolation, Modulation, Demodulation and Decimation **14**
4.1 Revisiting the physical channel . . . 15
4.2 Interpolation . . . 15
4.3 Modulation . . . 17
4.4 Transmission over channel, synchronization . . . 17
4.5 Demodulation . . . 17
4.6 Decimation . . . 18
4.7 Tasks part B . . . 18
4.8 Report . . . 19

1 Administrative details

1.1 Purpose

The goal of the projects is to give the students an opportunity to get some hands-on experience on using some of the signal processing tools and algorithms that are covered in the course. The projects are intended to be inspirational, and the students are encouraged to play around with their implemented systems to explore the features of the systems and make connections to the theories. The project work is conducted in groups of four students.

1.2 Project examination

The work of each project group is evaluated through the submitted report and an oral exam. To pass, the group has to show the unique passphrase displayed by MATLAB when all coding tasks have been completed as well a a basic understanding of the communication scheme. To receive higher grades, the groups also need to show understanding of the system features and connect to the theories covered in the course.

Report

Reports for part A and part B are to be uploaded in pingpong. Do not let the report exceed five pages. This limit does not include the cover page, pages with figures, and the appendix. It is OK (preferable even) to interleave figures and text as long as the total length of text does not exceed 5 pages with all figures removed. The title-page should contain the project number, group name, student names, the secret passphrase generated by MATLAB, and the associated student's birthdate used to generate the passphrase.

Oral exam

In the oral exam course staff will discuss different aspects of the project with the students. It is an opportunity for the students to ask questions to learn more, and for the staff to sense the level of understanding in the group. The oral exam is a 20-minute session. An individual pass/fail grade (0/6 points) is given to each student.

Grades

The project part of the course can generate maximum 36 points, distributed as follows:

Project	Item	Maximum points
1A	Report	8
1B	Report	8
1	Oral exam	6

1.3 Deadlines and oral examination

The deadline for the project reports and instructions for how to sign up for the oral examination can be found on the associated page in pingpong.

2 Project 1 overview

2.1 Introduction

In many practical scenarios including digital communications and image restoration, we measure a desired signal that has been passed through a linear system (which we can equivalently regard as a linear filter). In order to simplify further processing it is often desirable to restore the original signal as much as possible. When using a linear filter,

this operation is often called inverse filtering, deconvolution or equalization. In this project we are going to design a data transmission system over an acoustic channel, transferring text messages between two units via sound in the air. The project addresses the following problems:

1. signal encoding/decoding from the bit stream to symbols
2. conversion to an analogue coding/decoding technique; orthogonal frequency division multiplexing (OFDM)
3. transmission channel equalization
4. modulation/demodulation of the signal to/from a frequency band suited for acoustic transmission
5. interpolation/decimation to convert the signal between different sampling rates in the system

2.2 Tasks in Part A and Part B

The project is divided into two parts (part A and part B). Part A handles problems 1-3 above while part B adds problems 4-5. Part A is studied in MATLAB only. In part B you will implement the full system both in MATLAB and on the DSP-kit. This means you will convert the algorithms used in part A into C-code suitable for the DSP-board. For both the Matlab and C-code we provide a basic code structure with most parts completed, where your task is to complete the missing key parts in various functions.

In both parts it is generally a good idea to fully implement the missing code in the `student_sols.m` file, checking the correctness of your code by executing `proj1a.m` or `proj1b.m` respectively. Note that the self-test function will test functions for both part A and part B, so while evaluating part A the unfinished functions in part B will continue to generate `FAIL` statements.

3 Part A: Baseband Communication

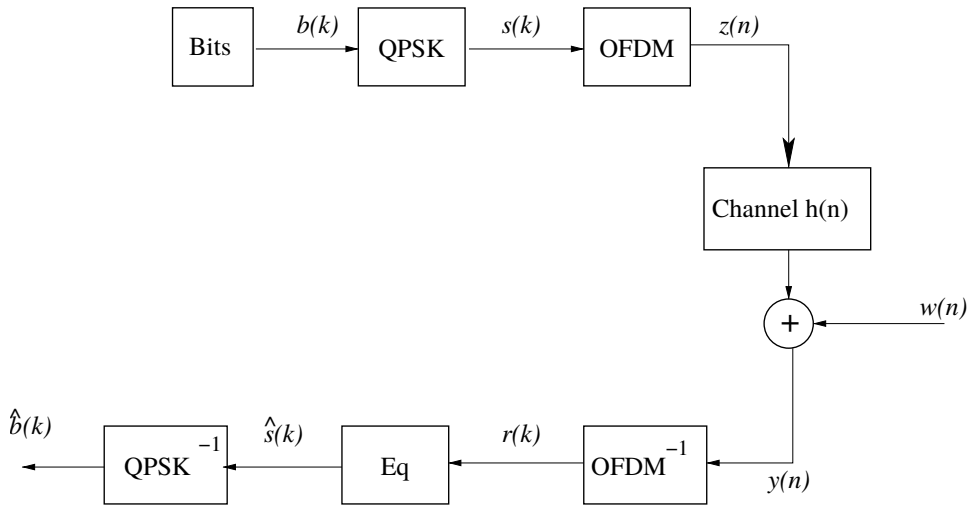


Figure 1: Block diagram for part A.

Figure 1 illustrates a block diagram of the steps taken in part A of the project. The following subsections will describe each of the listed blocks in more detail.

3.1 QPSK coding

The goal of our communication protocol is to transmit useful data bits over the channel. Specifically, assume we want to transmit a bit sequence of length $2N$.

$$b(t) \in \{-1, 1\}, \quad t = 0, \ldots, 2N-1$$

We will represent this sequence of bits using a sequence of *symbols*, here using *quadrature phase shift keying* (QPSK). In QPSK, the k'th symbol $s(k)$ is given by

$$s(k) = \sqrt{\frac{1}{2}}(b(2k) + jb(2k+1)), \quad k = 0, \ldots, N-1. \tag{1}$$

Hence each symbol carries the information of two binary bits. We will later see that the scaling term $\sqrt{\frac{1}{2}}$ is convenient as this ensures that all symbols have unit magnitude (magnitude 1). Each symbol has the following four possible values:

$$s(k) = \begin{cases} \sqrt{1/2}(1+j), & b(2k) = 1, \quad b(2k+1) = 1 \\ \sqrt{1/2}(-1+j), & b(2k) = -1, \quad b(2k+1) = 1 \\ \sqrt{1/2}(1-j), & b(2k) = 1, \quad b(2k+1) = -1 \\ \sqrt{1/2}(-1-j), & b(2k) = -1, \quad b(2k+1) = -1 \end{cases} \tag{2}$$

3.2 The physical channel

The acoustic system will alter the transmitted information. Hence, to be able to retrieve the information from the received signal the effect of the acoustic system, i.e. the channel, must be taken into account. To succeed in this task both the transmitter as well as the receiver must be properly designed.

The following issues will need attention for a design of a successful system:

- The acoustic system can be modeled as a linear time invariant filter. As such the transmitted signal will be filtered and the received signal will be the convolution of the transmitted signal and the impulse response of the channel. However, as the channel is unknown we need to devise a method which allows us to estimate the effect of the channel and then compensate for it in order to restore the information. This is known as *equalization*.

- The acoustic system has a bandpass character with a small gain for low and high frequencies. This is primarily due to the loudspeakers (limited to signals $> 50\,\mathrm{Hz}$) and microphone (limited to signals $< 8\,\mathrm{kHz}$) used. Because of this the communication system must send the information at frequencies where the acoustic system has a good gain. On the transmitter side this will involve signal interpolation and modulation and on the receiver side demodulation and decimation. These steps will be considered only in part B of this project.

- Since the transmitter and receiver work asynchronously with respect to each other the reception of a message must be detected. This is known as synchronization and must be performed on the receiving side. This will primarily be of concern when the DSP-kits are used, as the MATLAB environment is only a simulation and can "cheat" to know when the transmitter sends data.

3.2.1 Channel model

In this part of the project we will simulate the transmission of our signal in the *baseband*. This implies that we will send data in a range of frequencies that includes the zero frequency, i.e. a constant value. As described earlier, the physical audio channel is not suitable for this so we will use other simulated channels for this part of the project.

We will consider several different channel models in this part of the project. Your system will be evaluated on all of them in different ways as described later.

Channel models

$$h_1(n) = \begin{cases} 1, & n = 0 \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

$$h_2(n) = \begin{cases} 0.5, & n = 0 \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

$$h_3(n) = \begin{cases} e^{\frac{j}{2}}, & n = 0 \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

$$h_4(n) = \begin{cases} 0.8^n, & n \in [0, 59] \\ 0, & \text{otherwise} \end{cases} \tag{6}$$

$$h_5(n) = \begin{cases} 0.5, & n = \{0, 8\} \\ 0, & \text{otherwise} \end{cases} \tag{7}$$

3.3 Time domain equalization

A straightforward approach to communication is simply to send the symbols over the channel, i.e. we interpret the symbols $s(k)$ as a time domain sequence $z(n) = s(n)$ which we send over the channel (this of course assumes the channel can transmit complex values). The signal on the receiver end is thus

$$y(n) = z(n) \star h(n) + w(n) = \sum_{k=0}^{N_h-1} h(k)z(n-k) + w(n) \tag{8}$$

where $y(n)$ is the received signal and $w(n)$ denotes noise which the channel adds to the signal. Due to the dynamic effect of the channel (i.e. $y(n)$ depends on previous $z(n)$) the received signal $y(n)$ will be a mix of the transmitted symbols. This is known as inter symbol interference (ISI). If the channel is known, (i.e. $h(k)$ is known) the ISI can be removed/reduced by an equalization step which in principle means to filter the received signal through an approximate inverse channel. Such an approach can be regarded as a time domain equalization. However in this project we will use a frequency domain equalization method instead, since it fits so well with the chosen analogue coding technique, OFDM.

3.4 OFDM based communication

A sequence of discrete symbols $s(k)$ needs a conversion to an analogue representation if it is to be sent over a physical medium. One such technique is Orthogonal Frequency Division Multiplexing (OFDM) which sends a number of symbols in parallel using orthogonal sub-carriers (i.e. frequencies), one carrier for each transmitted symbol $s(k)$. With the QPSK modulation we are using we can intuitively view this as transmitting many frequencies simultaneously, where the phase of each frequency is determined by our symbols $s(k)$. In general, an OFDM signal with N information symbols has the

following form on the transmitter side

$$z(n) = \frac{1}{N}\sum_{k=0}^{N-1} s(k)e^{j2\pi kn/N} \tag{9}$$

where n is the sample index. Note that the OFDM sequence $z(n)$ is periodic with period N samples, the number of symbols transmitted in the package.

As an aside, note that $z(n)$ is complex-valued. Might this be an issue when we want to later transmit over the audio channel, which is by its nature a scalar, real-valued channel (air pressure being a scalar real value)? This is a topic we will return to in the part B of the project. For now, let's assume our channel can accept/propagate a complex-valued signal.

We directly notice that $z(n)$ given in (9) is indeed the inverse discrete Fourier transform of the sequence $s(k)$, the information symbols. Hence, trivially we can recover our information symbols $s(k)$, $k = 0, \ldots, N-1$ by calculating the DFT based on $z(n)$ $n = 0, \ldots, N-1$. This is an important property of the OFDM which makes the coding into and from OFDM use basic discrete Fourier transform theory. We denote the sequence $z(n)$ for $n = 0, \ldots, N-1$ as one *OFDM block* .

If we, for the moment, assume that the transmitter was turned on at $n = -\infty$, the signal at the receiver end will have the form

$$y(n) = z(n) \star h(n) + w(n) = \cdots = \frac{1}{N}\sum_{k=0}^{N-1} H(k)s(k)e^{j2\pi kn/N} + w(n), \quad n = 0, \ldots, N-1 \tag{10}$$

where $H(k)$ is the discrete time Fourier transform at frequency $2\pi k/N$ of the impulse response of the channel:

$$H(k) = \sum_{n=0}^{N_h-1} h(n)e^{-j2\pi kn/N}, \quad k = 0, \ldots, N-1 \tag{11}$$

$H(k)$ is also called the (complex) channel gain at frequency index k.

The received signal $y(n)$ is a periodic signal with period N. Equation (10) reveals that the original sequence of symbols is *almost* extracted by using the discrete Fourier transform on one period of the sequence $y(n)$. The complex scaling $H(k)s(k)$ is consequently the coefficients of the Fourier series expansion and can be obtained using the DFT

$$r(k) = \sum_{n=0}^{N-1} y(n)e^{-j2\pi kn/N} = H(k)s(k) + n(k), \quad k = 0, \ldots, N-1 \tag{12}$$

where $n(k)$ denotes the noise in the frequency domain.

Crucially, if $H(k)$ is known or estimated in a previous step we can recover $s(k)$ simply by dividing $r(k)$ with $H(k)$.

3.5 Cyclic prefix

Our assumption that the transmitter was turned on at $n = -\infty$ is of course not realistic. However, since the channel was assumed to have a finite length impulse response the channel response will become stationary, i.e. periodic, after N_h samples. Hence if we start the transmitter at sample $n = -N_{cp}$ where $N_{cp} \geq N_h$, according to (9) the received signal at time $n = 0$ will be stationary. This is called adding a cyclic prefix. Note that the cyclic prefix is equal to the last N_{cp} samples of the $\{z(n)\}_{n=0}^{N-1}$ signal (consider why this is the case!). We denote the OFDM block with the added cyclic prefix in front as one *OFDM package*. In summary:

- The transmitter is appending a cyclic prefix of length N_{cp} to the front of the $z(n)$ signal ($n = 0, \ldots, N-1$)
- The receiver detects the start of the transmission (synchronization) and skips the first N_{cp} samples of the received signal. The following N samples are taken as the received OFDM message $y(n)$, $n = 0, \ldots, N-1$, and as $N_{cp} \geq N_h$, *we are guaranteed that $y(n)$ is identical to the case where the transmitter was turned on at $n = -\infty$!*

3.6 Equalization with known H(k)

The sequence $r(k)$ is the symbol sequence $s(k)$ modified by the channel gain $H(k)$. The next step is to remove the effect of the channel, a process known as equalization.

Division by the complex channel gain at frequency k results in

$$r(k)/H(k) = s(k) + n(k)/H(k). \tag{13}$$

As we have chosen to use QPSK symbols, we can decide on what symbol $s(k)$ was received by checking the real and imaginary parts of the quotient $r(k)/H(k)$, so long as the noise level is small enough. The equalization will possibly fail for some sub-channels if $|H(k)|$ is small or zero in comparison with the level of the noise energy at that sub channel, i.e. the SNR is low for frequency k.

Note that the equalization for QPSK symbols also can be performed by considering

$$H^*(k)r(k) = H^*(k)H(k)s(k) + H^*(k)n(k) = |H(k)|^2 s(k) + H^*(k)n(k) \tag{14}$$

which only requires a complex multiplication and is less computationally intensive compared to (13). We will use this in the DSP implementation.

If we know the channel $H(k)$ we can decode the $2N$ sent information bits as

$$\begin{aligned} \hat{b}(2k) &= \mathrm{sgn}(\mathrm{Re}\{r(k)/H(k)\}) \\ \hat{b}(2k+1) &= \mathrm{sgn}(\mathrm{Im}\{r(k)/H(k)\}) \end{aligned} \tag{15}$$

For visualization of the quality of the received signal it can be useful to also plot the "soft" symbol estimates. These are obtained by

$$\hat{s}(k) = r(k)/H(k) = H^*(k)r(k)/|H(k)|^2 \tag{16}$$

3.7 Equalization with unknown H(k)

If the channel impulse response is unknown some method must be employed to establish sufficient information about the channel. In this project we employ a very simple method. We communicate by sending an *OFDM frame* which contain two OFDM packages (OFDM blocks including cyclic prefixes) placed back to back. The first OFDM package contain N symbols which are pre-determined and known by both the receiver and transmitter. These are called *pilot* or *training* symbols. The second OFDM package will hold the useful data bits we want to transmit. As the receiver knows the transmitted pilot symbols in the first OFDM pilot package, the channel gain is estimated as follows

$$\hat{H}(k) = r_p(k)/s_p(k) \tag{17}$$

where $s_p(k)$ and $r_p(k)$ are the known and received values of the pre-determined pilot symbols respectively. The estimated channel gains are then used to recover the symbols $s_d(k)$ in the second, ultimately useful, OFDM data package.

Using this scheme (pilot + data packages) is fairly inefficient, as we need to send twice as much data. Some methods (that we will not further study) that can be used to reduce this inefficiency are:

- If the channel is time invariant during the duration of the OFDM frame, it is possible to use a larger frame where several OFDM data packages follow the OFDM pilot package. This would the reduce the overhead incurred by the pilot package and increase the number of transmitted bits per time unit.

- If $H(k)$ only varies slowly over frequency then we could decide to use some subcarriers (i.e. $s(k)$ for some indices k) to be used for pilot symbols and then estimate the channel gain ($H(k)$) for these frequencies. These channel gain estimates can then be used at nearby frequencies in the equalization since the difference between $H(k)$ and $H(k \pm m)$ will be small if m is small. This means that the information symbols are mixed with pilot symbols in the same OFDM frame.

- We could use the fact that all symbols transmitted only take four distinct complex values. If we correctly decoded symbol $s(k)$ we can estimate $H(k)$ as simply as $\hat{H}(k) = r(k)/\hat{s}(k)$. We can then use $\hat{H}(k)$ as the channel estimate for $H(k+1)$ and so on. In this way we only need to transmit a few pilot symbols to have a baseline to start from. This type of methodology is known as *decision feedback* since the output from the detector is fed back into the channel equalization. Might this method fail in some cases?

3.8 Tasks Part A

Develop your code according to the instructions in `proj1a.m`. Be sure to include the secret passphrase and the `student_id` parameter used on the cover page of your report. When you have verified correct code you are ready to perform some experiments where the system operates under non-ideal conditions.

Note that `sim_ofdm_known_channel` and `sim_ofdm_unknown_channel` will *not* be self-tested, it is up to you to study the results and determine if the function is correct.

There are some parameters in `proj1a.m` which can be varied to vary the problem set-up:

- `h` The impulse response of the channel to use.
- `N_cp` The length of cyclic prefix to use
- `snr` The effective signal to noise ratio (SNR) at the receiver side. The SNR is defined as signal power divided by noise power and is given in dB as SNR $= 10 * \log_{10} \frac{P_s}{P_n}$. Suitable values for SNR are between 0 and 50 dB. If you want a noise free transmission set `snr = inf` (infinite SNR).
- `sync_err` Up to this point we have assumed that the receiver correctly starts "listening" at the right point in time. In practice, we are almost ensured that there will be some offset in time, i.e. instead of receiving the signal $y(n)$ we will receive $y(n+n_{\text{sync_err}})$. The `sync_err` parameter allows for setting this offset in time (an integer number of samples), where positive values correspond to the receiver taking a block of data that is too late, while negative values correspond to data that is too early.
- `known_channel` With this boolean value you can select whether to use the code you've written for the known-channel case or the unknown-channel case.

The function also automatically provide you with some performance metrics:

- `ber` The bit error rate. The number of incorrectly received bits divided by the total number of transmitted bits.
- `evm` The average error vector magnitude. The error vector magnitude is the distance between the correct QPSK symbol and the soft decoded symbol $r(k)/H(k)$.

Investigate the following scenarios and include your findings in your report.

1. Start with the default case: The ideal (trivial) channel h_1 is used, no noise is added (`snr = inf`, i.e. $w(n) = 0$), the cyclic prefix is set to zero, there is no syncronization error, and the channel is known. *We will repeatedly return to this setup, so make a note as to these settings.* (The choice of a channel without any dynamics and no noise implies that $y(n) = z(n)$ in (10).)

(a) Investigate and consider how this setup affects the received symbols, both before ($r(k)$) and after ($\hat{s}(k)$) equalization. (See the plots that are generated in the `proj1a.m` script). How are the transmitted symbols related to the received symbols? What is the EVM and the BER? Why?

(b) What influence does the cyclic prefix have on the ability of our communication scheme to work? How long does the cyclic prefix need to be to give a near-zero EVM? Why?

(c) Now, back to zero cyclic prefix, change the channel to h_2 (magnitude-scaling) and h_3 (constant phase-shift). Notably, with these channels we are ensured that $y(n) = \alpha \cdot z(n)$, for a scalar, possibly complex, value α.

How does this choice of channel affect the pre- and post-equalization constellation diagrams, the EVM, and the BER? What is α in both of these cases?

(d) Now we will consider the effect of synchronization error. Return back to the initial setup. Initially test a synchronization error of ± 1. What happens when it is increased? Why does this have such a large influence on the EVM and BER?

Hint: Study the received text string when adding synchronization error. Note that the start and end of the message (corresponding to the first and last bits in the transmitted sequence) are correctly transmitted for ± 1, while the middle of the message is completely mangled. When the synchronization error is increased to ± 2, we instead have two mangled sections while the start, middle, and end are correctly received. Which subcarrier frequencies do the mangled and correct regions correspond to? To what degree are these subcarriers phase-shifted when adding a sync error? What determines whether a given symbol is converted by the receiver into the correct (transmitted) symbol or not? Recall that the synchronization error can be viewed as an unexpected delay/early reception of the signal, which in the frequency domain is equivalent to multiplication by $e^{-j \cdot \text{sync_err} \cdot f_{\text{normalized}}}$.

(e) Finally return to the initial setup and start adding some noise to the system, e.g. set `snr = 30`. What happens, and why? What about for `snr = 5`?

2. Let us now progress to (slightly) more realistic scenarios. Return to the initial setup, but change the channel model to h_4 (the low-pass system) and set the cyclic prefix to 60. Notably, this system does have dynamics, so generally, $y(n) \neq \alpha \cdot z(n)$.

(a) How are $H(k)$ and the pre-equalization constellation diagram related? Your answer to this question can be fairly brief and based only on visual observation of the plots of $H(k)$ and the pre-equalization constellation diagram.

Hint: Note that $r(k) = H(k) \cdot z(k)$. As $H(1) = 5$, will we find one symbol at $5 \cdot \sqrt{\frac{1}{2}}(\pm 1 \pm j)$?

From now on, we will only consider the post-equalization symbols.

(b) What is the BER and EVM of the system with this setup? What happens when N_{cp} is increased/decreased? Is there some "magic" number that makes EVM $= 0$ (or at least $< 10^{-15}$, which is within the machine accuracy of zero)? Why is it this particular value? How is the BER influenced by the choice of N_{cp}?

(c) Modify the channel definition to $h'_4 = 0.99^n$. How important is the choice of N_{cp} with this channel, both with respect to the BER and EVM?

3. Now, let us move to the fairly realistic case: return to the initial setup between each of these subquestions, but keep `channel_known = false;`

   (a) Select the ideal channel h_1, set N_{cp} to zero and investigate the effect of nonzero sync error. How and why is this different compared to the known-channel scenario?

   (b) Select the lowpass channel h_4. How large does the cyclic prefix need to be to give a near-zero EVM? Is this the same value as before? Why? Now, keep the cyclic prefix at a large enough value and change the SNR. How does the noise level influence the BER and EVM? How sensitive is this setup to noise compared to the known-channel scenario? Why?

   (c) Select the multi-path channel h_5. Set N_{cp} to a large enough value (which value is this, and why?). Why do we always have some bit errors despite the absence of any noise and sync error?

   Hint: test the modified channel $h'_5 = [0.5, 0, 0, 0, 0, 0, 0, 0, 0.4, 0, \dots]$ *(i.e. the second nonzero coefficient is set to 0.4 instead of 0.5). This will give a different behavior. Is there any critical difference between* $|H_5|$ *and* $|H'_5|$*?*

   (d) Finally, play around with N_{cp}, sync error, the SNR, and the choice of channels. Would you say that our communication scheme is robust, or sensitive to disturbances? Why?

4 Part B: Interpolation, Modulation, Demodulation and Decimation

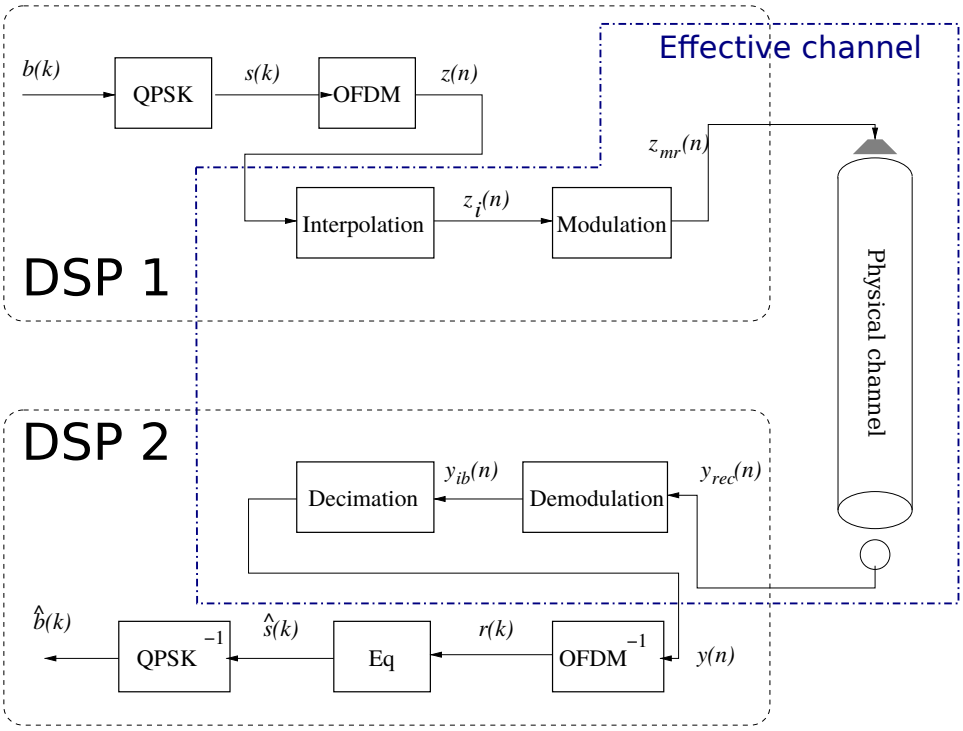


Figure 2: Block diagram for part B.

Figure 2 illustrates a block diagram of the system in part B of the project (compare with Figure 1). The following subsections will describe each of the new and modified blocks in more detail. Notably, we can view all of the parts within the dot-dashed blue line in as the "channel" the signal is sent through. This consists of the physical channel (i.e. sound through air), as well as any dynamics in the loudspeaker/microphone, and the interpolation, modulation, demodulation, and decimation steps. These parts can together be thought of a an effective channel, which for the noise-free case we can lump together and view as the channel h in Figure 1.

First, we will simulate the effects of performing interpolation, modulation, demodulation, and decimation in Matlab. Afterwards, we will test OFDM "in the wild" on the DSP board and try to send short text messages.

The physical system we are going to use consists of the DSP-board and a set of loudspeakers which are connected to the board. A microphone is mounted directly on the DSP-board. One board can be used as a transmitter for messages while a second board can act as a receiver. For development purposes a single board can act as both transmitter and receiver.

4.1 Revisiting the physical channel

In the previous part of the project, we simplified the project by assuming that we can transmit data in the baseband (i.e. down to 0 Hz). However, remember that transmitting at 0 Hz over an audio channel corresponds to controlling the absolute pressure of the ambient air, which would require the an impractical airtight sealed room and a loudspeaker with a large *displacement*. Our loudspeaker, with a diameter of 40 mm and a stroke of 2 mm displaces nearly no air relative to the volume of a normal-sized room, and is practically limited to signals ≥ 50 Hz. Furthermore, as the DSP-kit runs at a sample-rate of 16 kHz, the Nyquist sampling theorem limits the bandwidth of the signal to 8 kHz. One reasonable choice is to therefore require the transmitted signal in the physical channel to lie in the range $50 \leq f \leq 8000$ Hz.

In this part of the project we will use interpolation and modulation to move the OFDM signal to a range of frequencies that is more suitable. Apart from these aspects, the core OFDM structure will be otherwise unchanged from part A.

4.2 Interpolation

Recall that in part A of the project we created the baseband signal $z(n)$. With ideal *interpolation* we can increase the sample-rate of the signal without otherwise changing the signal. Assume our goal is to increase the sample-rate of the signal by an integer factor R. In an ideal setting, we can achieve this by up-sampling $z(n)$ by R followed by an ideal low-pass filter with normalized cutoff frequency $1/(2R)$.

Assume $z(n)$ is the original signal with spectrum $Z(\omega)$. The factor R up-sampler is defined as

$$z_u(n_u) = \begin{cases} z(n_u/R), & n_u = 0, \pm R, \pm 2R, \ldots \\ 0 & \text{otherwise,} \end{cases} \tag{18}$$

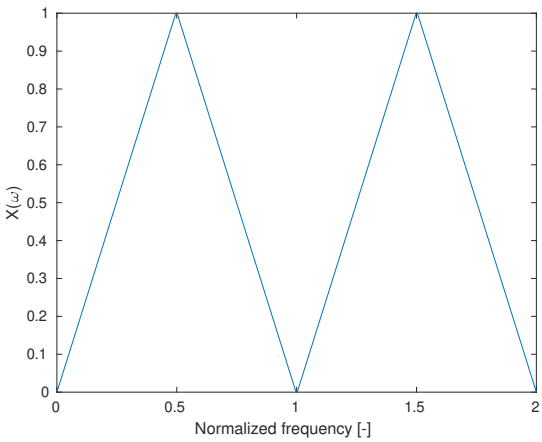
i.e. we place $R-1$ zeros between each element in z.

Interestingly, the spectrum for the up-sampled signal is simply

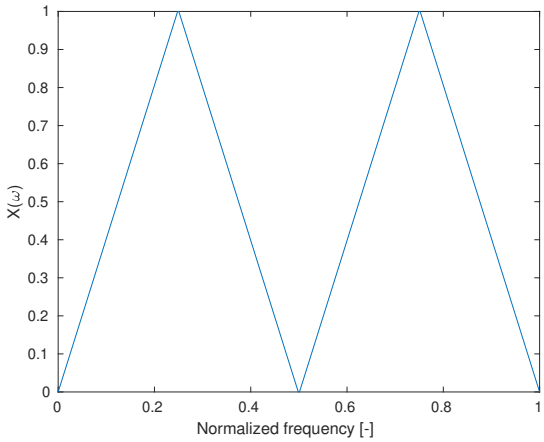
$$Z_u(\omega) = Z(\omega) \tag{19}$$

i.e the signal's spectrum is identical! (See Figures 3a and 3b). Remember though that as Z and Z_u are periodic with a period ω_s, the up-sampled spectrum will be composed of the original spectrum and a total of $R-1$ copies of the original spectrum within $0 < \omega < R\omega_s$.

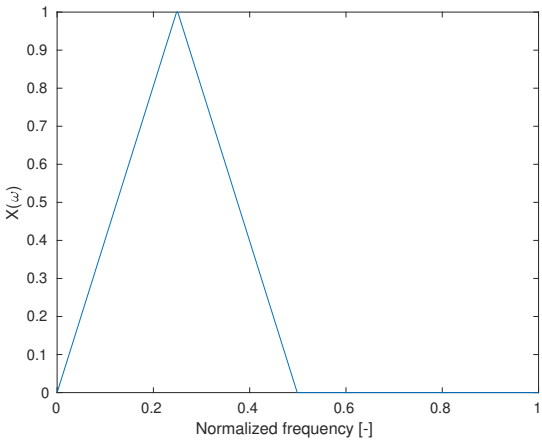
We can use a low-pass filter to remove the $R-1$ images of the original spectrum in Z_u, see Figure 3c. In practice, we will need to design a suitable lowpass filter that is an acceptable compromise of the "usual" parameters (pass-band flatness, stop-band attenuation, transition-band width, filter length, etc).



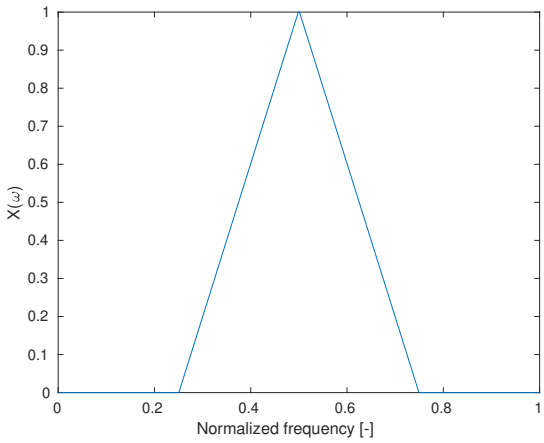
(a) Original signal, covers range $[0, f_s/2]$



(b) Upsampled signal, $R = 2$. Identical spectrum, note change in normalized frequency scale!



(c) Interpolated (upsampled + filtered) signal. Note how $R-1$ images are removed in range $[0, f_s/2]$.



(d) Interpolated, modulated signal. Here, shown for normalized modulation frequency $f_{cm} = 1/2$.

4.3 Modulation

With modulation we can shift the absolute position of a signal in the frequency domain. This is done by multiplying each sample with a complex exponential with frequency f_{cm}, the center modulation frequency. If $z_i(n)$ is the original signal, then modulated signal is given by

$$z_m(n) = z_i(n)e^{j2\pi n f_{cm}/f_s}, \quad n = 0, \pm 1, \pm 2, \ldots \tag{20}$$

The effect of this in the frequency domain is

$$Z_m(\omega) = Z_i(\omega - \omega_s f_{cm}/f_s) \tag{21}$$

which is also illustrated in Figure 3d.

4.4 Transmission over channel, synchronization

With the ability to interpolate and modulate our signal we can ultimately do something useful: namely move our OFDM signal to a range of frequencies that we can transmit through air. Of course, once we receive the signal we must move the signal from a range centered on f_{cm} to centered on 0 (i.e. move the signal back to the baseband), and reduce the sample rate back to the original sample rate, a description of which follows.

However, before we can consider the receiver, note that our signals have so far been free to lie in the complex domain. However, our physical channel (i.e. air pressure) can only carry real-valued signals. (As air pressure is a real-valued, not complex-valued, scalar field). In this project, we will manage this simply by discarding the imaginary part of the signal. As it turns out (under some conditions that you will later look at), discarding the imaginary part will not result in any loss of information.

Finally, the receiver needs to detect when an OFDM frame starts. This is known as synchronization. The DSP-kit has a synchronization method that uses the periodicity of the transmitted OFDM message to detect the signal. (Remember, adding the cyclic prefix makes the entire message periodic in the time domain, with period N.) As you will see, when working with the DSP-kit, the cyclic prefix must be selected to be long enough. In the Matlab simulation the start of the OFDM message is delivered as an output argument, so we can "cheat" and avoid the synchronization problem entirely.

4.5 Demodulation

Note that in (21), we can see that we can demodulate the signal by modulating the signal once more with the carrier $-f_{cm}$, which will bring our signal back to the base-band. Essentially, demodulation can be viewed as modulation with a sign-reversed carrier. Demodulation can be viewed as going from Figure 3d to Figure 3c.

4.6 Decimation

Our final step is to reduce the sample rate of the demodulated signal. For reasons you will consider in your report, we must first apply an anti-aliasing lowpass filter to the demodulated signal. As it turns out, a good choice is to re-use the same filter as we used in the interpolation step.

We can now reduce the sample-rate of a band-limited signal by a factor D by only keeping every D'th sample of the original signal and discarding all other samples. As our goal is to return to the original sample-rate, we should select $D = R$. In the time domain;

$$x_d(n) = x(nD), \quad n = 0, \pm 1, \pm 2, \ldots \tag{22}$$

In the frequency domain, this corresponds to

$$X_d(\omega) = \sum_{k=0}^{D-1} X(\omega + \frac{k\omega_s}{D}). \tag{23}$$

where ω_s in rad/s is the sample rate for the original signal $x(n)$. This corresponds to moving from Figure 3c to Figure 3a.

At this stage, i.e. after interpolating, modulating, transmitting, demodulating, and decimating we will have a signal $y(n)$ that is related to $z(n)$, and with ideal steps we can view $Z(\omega) = H_{\text{effective}}(\omega) \cdot Y(\omega)$ where $H_{\text{effective}}$ an effective channel given by all the above steps. Importantly, with the methods used in part A of this project we can estimate $H_{\text{effective}}$, allowing for us to transmit our OFDM message over a physically realistic channel.

4.7 Tasks part B

Matlab implementation

Develop your code according to the instructions in `projba.m`. Be sure to include the secret passphrase and the `student_id` parameter used on the cover page of your report. When you have verified correct code you are ready to perform some experiments.

Note that `sim_ofdm_audio_channel` will *not* be self-tested, it is up to you to study the results and determine if the function is correct.

DSP-kit implementation

All source files are located in the `src` directory. You will need to add code to several functions in `lab_ofdm_process.c`. These functions are called from the two main functions:

- `lab_ofdm_process_tx()` which assembles the OFDM frame to be transmitted
- `lab_ofdm_process_rx()` which decodes the received OFDM frame.

Specifically you need to add code to the functions:

- `ofdm_demodulate()`
- `cnvt_re_im_2_cmplx()`
- `ofdm_conj_equalize()`

Sections that require work on your behalf are indicated with `/* TODO: Add code from here... */` and `/* ...to here */`. Read the comments given in the functions to get hints on what the added code should perform. Note that the two first functions have an "inverse" counterpart used when assembling the OFDM frame to be transmitted which you can look at for inspiration.

Remember that complex numbers are not available in C. You need to explicitly keep track of the real and complex parts as two separate real-valued numbers (using the `float` datatype). A complex vector of length n will therefore require $2n$ floating point variables, which together represent the the complex vector.

4.8 Report

The questions in this part of the project are intentionally more open-ended. It will be up to you to determine both how to change the system setup to test the asked behavior and to interpret the results.

In the report you should briefly answer the following questions and explain your reasoning:

- Based on the Matlab implementation
  1. Assume the baseband signal is interpolated by a factor of 8, a modulation frequency 4 kHz is used, and end up at a final sampling frequency of 16 kHz. What is the bandwidth of the transmitted audio signal and between which frequencies does the transmission band lie?
  2. In part A we could acheive an EVM which was nearly zero if we set `snr=inf`. However, if we try to do the same thing with our new setup, we always get a nonzero EVM. Why is this the case? Note: this is *not* due to any non-ideality of the channel, as if we modify `simulate_audio_channel.m` and add `h=1` to replace the simulated audio channel (after `h = impz(b, a);`) we will still see the same behavior.
  3. In the receiver, H is estimated from the pilot OFDM block. What parts of the system (i.e. interpolation, modulation, taking the real part of the signal, propagation over the physical channel, demodulation, decimation) contribute to the channel H? How do they contribute?
  4. Before transmission over the physical channel we have chosen to simply discard the imaginary part of the complex-valued signal. As it turns out, we could also have chosen to discard the real part and keep the imaginary part of the signal

(or, generally, keep an arbitrary projection of the signal onto $\cos(\alpha) + j\sin(\alpha)$ for any α.). Explain why (and in general when) we can safely discard either the imaginary or the real part of the signal. You do not need to show why this holds in the general case. *Hint: if z is the signal we wish to transmit, then discarding the imaginary part gives $z_r = \frac{1}{2}(z + \bar{z})$, where $\bar{z}$ is the complex conjugate of z. Furthermore, due to the linearity of the Fourier transform, we are also ensured that $Z_r(\omega) = \frac{1}{2}(Z(\omega) + \bar{Z}(-\omega))$.*

5. In the interpolation and decimation stages, which of the following lowpass filter properties are more important, and which are less important? Why? Specify your answers separately for the interpolation and decimation stages.
   - Passband ripple
   - Stopband attenuation
   - Transition band width
   - Phase linearity

- Based on tests using the DSP-kit. Let T, H, and R be the Fourier transforms of the transmitted signal, channel, and received signal respectively. The notation $\bar{x}$ will be used to indicate the complex conjugate of x.

7. The channel estimate is generated as $\hat{H} = \bar{T} \cdot R$. We avoid using the division operation because it is computationally much more demanding than performing a complex conjugation and multiplication. (Most computers generally perform some type of long division similar to the kind you learned in grade school, while a multiplication is performed in one clock cycle). Assuming an ideal system, show that $\hat{H}$ and H have the same phase *and* magnitude.
8. The received symbols are equalized as $R_{\text{eq}} = R \cdot \bar{\hat{H}}$. Show that this correctly equalizes the phase of R_{eq}.
9. With a constant background noise level and physical setup, test sending messages with high and low signal amplitudes. (Use the serial monitor to change the amplitude in real-time). How does the EVM and constellation diagram change? Why?
10. Hold the transmitting speaker in your hand, above and directed towards the DSP kit. Start a transmission, and start moving the speaker towards the DSP kit at a speed of around 2-5 cm/s. The goal is for the speaker to be moving at a constant speed during the short time when the signal is transmitted without it crashing into the DSP-kit before the transmission is complete. How does this motion change the received symbols, especially with respect to the constellation diagram? What about if you instead start near the DSP-kit and then move away from it? Why do you get this specific behavior? *Hint: remember that moving the speaker implies that the channel changes over time, i.e. the channel during the time the training symbols are sent is different when compared to the data symbols.* Can you estimate how quickly you were moving the speaker based on the constellation diagram? What speed would theory

suggest is the highest speed that doesn't give any bit errors (your answer need only be correct to an order of magnitude)?