NTNU

Innovation and Creativity

Underwater Communications

An OFDM-system for Underwater Communications

Svein Erik Søndervik Gregersen

Master of Science in Electronics

Submission date:	June 2007
Supervisor:	Jens Martin Hovem, IET
Co-supervisor:	Tor Arne Reinen, SINTEF IKT

Norwegian University of Science and Technology
Department of Electronics and Telecommunications

Problem Description

Define and implement a system for underwater communications using OFDM and DQPSK, and perform initial measurements based on the system.

Assignment given: 15. January 2007
Supervisor: Jens Martin Hovem, IET

An OFDM-system for Underwater Communications

Gregersen, Svein Erik

June 5, 2007

Abstract

In the fall 2006 NTNU (The Norwegian University and Science and Technology) initiated a strategic project in cooperations with SINTEF where the aim is to gain more knowledge about underwater acoustic communications. This study is a part of this project and focuses on a system for underwater communication. A orthogonal frequency division multiplexing (OFDM) system using differential quadrature phase shift keying (DQPSK) has been defined and implemented in MATLAB. The system has been characterized through thorough simulations and testing. Initial measurements has also been carried out in order to test the developed system on a real underwater acoustic channel and the results have been analysed.

Preface

This master thesis in the subject advanced marine acoustics has been written by Svein Erik Gregersen during the spring semester of 2007 at The Norwegian University of Science and Technology (NTNU). In discussion with Tor Arne Reinen at SINTEF and Jens M. Hovem at NTNU the subject of a underwater communication system was proposed. I will like to thank Tor Arne Reinen for the cooperation during the spring and for helping me find the answer to numerous questions. I would also like to thank Jan Erik Håkegård and Knut Grythe for their help during the development of the OFDM-system. Last I would like to thank Jens M. Hovem for guidance and inspiration during my study.

Trondheim, June 5, 2007

Svein Erik Gregersen

Contents

1 Introduction **1**

2 Theory **3**

2.1 Communications theory 3

2.1.1 OFDM 3

2.1.2 QPSK 7

2.1.3 Differential-QPSK 8

2.2 Environmental conditions and physical properties 8

2.2.1 Geometrical spreading and absorption 9

2.2.2 Sound speed profile 10

2.2.3 Doppler spread 11

2.2.4 Bottom interactions 11

3 Communication system **13**

3.1 The system 13

3.1.1 The transmitter 14

3.1.2 The receiver 15

3.2 Implementation 16

3.2.1 The transmitter 16

3.2.2 The receiver 19

4 Simulations **23**

4.1 OFDM simulation 23

4.1.1 Transmission 24

4.1.2 Reception 28

4.2 Channel simulations 31

4.2.1 AWGN 31

4.2.2 Phase error 34

4.2.3 Multipaths 37

4.2.4 Synchronization error 40

4.2.5 A ”real” communication channel 41

5 Initial measurements **45**
5.1 The area of interest 45
5.1.1 Locations 45
5.1.2 The seabottom 46
5.1.3 Environment 47
5.2 Hardware setup 48
5.2.1 Transmitter 48
5.2.2 Receiver 50
5.2.3 Signal format 50
5.3 Results 51

6 Discussion and conclusions **61**
6.1 The system 61
6.2 Initial measurements 62

A Bore hole data from NGU **67**

B Measurments **69**

C Matlab-code **71**

List of Figures

2.1 Spectrum of OFDM sub-carriers [9]. 4
2.2 A simple OFDM system [6]. 5
2.3 Intersymbol interference: the green symbol was transmitted first, followed by the blue symbol [6]. 5
2.4 An example of a QPSK signal-space 7
2.5 The attenuation in seawater as a function of frequency 9
2.6 Typical sound speed profiles for summer and winter. 10

3.1 Block diagram of the transmitter. 14
3.2 Block diagram of the receiver. 15
3.3 An overview of the implementation of the transmitter. 16
3.4 Illustration of differential encoding. 18
3.5 Block diagram of the receiver. 19

4.1 The transmitter. 24
4.2 The wave-file and the bit stream *DataIn*. 24
4.3 Plot of the first 12 QPSK symbols to be transmitted. 25
4.4 The mapped complex phase differences. 25
4.5 The frequency spectrum at the base band. 26
4.6 The frequency spectrum after the upshift to the carrier frequency. 27
4.7 The signal to be transmitted over the channel. 27
4.8 The receiver. 28
4.9 The frequency spectrum of the received signal after downshift to baseband. 28
4.10 The complex phase differences at the receiver. 29
4.11 The received QPSK symbols at the receiver. 30
4.12 The first 24 received bits at the receiver and the resulting wave-file. 30
4.13 Illustration of an underwater channel 31
4.14 The received signal with and without noise. 32
4.15 The received signal with and without noise. 32
4.16 The complex phase difference with noise. 33
4.17 The received wave-file with a 18 dB SNR. 33

4.18 The bit error ratio as a function of SNR. 34
4.19 Phase error as a function of time. 35
4.20 The complex phase difference with phase error. 35
4.21 The received wave-file with phase error. 36
4.22 The bit error ratio as a function of SNR. 36
4.23 The channel impulse response. 37
4.24 The transmitted signal and the received signal. 38
4.25 The phase difference with multipaths. 38
4.26 The received wave-file with multipaths. 39
4.27 The bit error ratio as a function of SNR. 39
4.28 The phase differences with FFT-window offset. 40
4.29 The bit error ratio as a function of offset in time. 41
4.30 The received signal after channel influence. 41
4.31 The calculated phase difference after channel influence. 42
4.32 The retrieved wave-file at the receiver. 43
4.33 Bit error ratio for the communication channel. 43

5.1 Trondheim Harbor 45
5.2 Profile of the seabottom 46
5.3 A 3D map of the bottom 46
5.4 Sound speed profile measured 17.11.06 47
5.5 A ray trace with the sound speed profile a position 2. 48
5.6 Left: The setup for the experiment. Right: Two transducers mounted on the aluminum rack at the receiver. 49
5.7 Directivity pattern for the transducer 49
5.8 Screen shot of one of the experiment signals (12 dB/div in the frequency plot) 51
5.9 The channel impulse response. 52
5.10 The transmitted signal (6 dB/div in the frequency plot). . . . 53
5.11 Received signal at "Sjøbadet" (6 dB/div in the frequency plot). 53
5.12 One transmission received at "Sjøbadet" (6 dB/div in the frequency plot). 54
5.13 The correlation function of the reference symbol and the received signal. 55
5.14 The calculated complex phase difference of the received signal. 56
5.15 The received signal in the simple experiment (6 dB/div in the frequency plot). 57
5.16 The calculated complex phase difference of the received direct signal. 57
5.17 The simulated oscillator drift. 58
5.18 The calculated complex phase difference with oscillator drift correction. 59

List of Tables

2.1 Signal-space characterization of QPSK 8

3.1 QPSK-symbol mapping 17
3.2 Phase shift mapping 17

4.1 Numerical values for the OFDM parameters. 23

A.1 Data from the boreholes 67

B.1 The experiment signals 69

Chapter 1

Introduction

Underwater communication between nodes using acoustic waves is becoming more and more relevant in both autonomous underwater vehicles (AUV) and static surveillance systems. The applications are in estimation of fish resources, surveillance of oil and gas pipelines, environmental surveillance and the surveillance of harbor areas. For these area of applications there is a need for a large number of nodes working in a network.

Today there exists several solutions for underwater acoustic communication. These are robust point to point systems with a conservative design. Much can be gained by using modern communication- and network principals developed for land based systems. This is necessary to be able to implement large underwater network systems. The conversion of the technology from land based to subsea is not easy as the channel parameters are radically different. The result of the much lower speed of the acoustic wave underwater and the large number of mulitpaths is long impulse responses and large time delays. The underwater acoustic channel is also bandlimited due to a frequency dependent absorption in the water.

In the fall 2006 NTNU (The Norwegian University of Science and Technology) initiated a strategic project in cooperation with SINTEF. This project aims to gain more knowledge in the field of underwater channel modeling and communications. The background for this is that there is a need for a system for environmental surveillance of the different sea zones in Norway, especially in the northern zones. The study presented here is just a small part of this project.

This study is divided into two parts. The first and most important objective is to define and implement a system for underwater communications. Here we will use orthogonal frequency division multiplexing (OFDM) as a modulation technique because of its ability to handle multipaths. In order to reduce the systems complexity and to improve the data rate, differential quadrature phase shift keying (DQPSK) will be used as the coding technique. The system will be implemented in MATLAB and simulations will be carried out in order to characterize the OFDM-system. The aim is to develop a system that is able to function properly in the challenging conditions of a underwater acoustic channel.

The second objective in this study is to carry out initial measurements using the OFDM-system developed in the first part. The measurements will take place at Trondheim harbor. Here signals created by the OFDM-system will be transmitted underwater and analysed. Based on this analysis the OFDM-system will be evaluated.

Chapter 2

Theory

In this chapter we will look at some topics which are important in the field of underwater communications and for the understanding of the work done in this study.

2.1 Communications theory

In this section we will take a closer look at some topics concerning modern communication principles which is used in wireless systems. We will take a look at the principles of OFDM and the theory behind, plus give a brief description of QPSK and differential-QPSK

2.1.1 OFDM

In wireless communications efficient and reliable transmission of information signals over challenging channels is a central issue. One way to achieve this is by using multi carrier modulation (MCM). The principle of MCM is to divide the transmission channel into a number of sub-channels or sub-carriers. Orthogonal frequency division multiplexing (OFDM) is a form of MCM where its carrier spacing is chosen so that each sub-carrier is orthogonal to the other sub-carriers. By orthogonal we mean that when the dot product of two deterministic signals is equal to zero, these signals are said to be orthogonal to each other. This will allow the sub-carriers spectra to overlap, and thereby increasing the spectral efficiency [6]. In OFDM the entire channel bandwidth is divided into several sub-bands as shown in figure 2.1. This gives a relatively flat frequency response over each individual sub-band. The systems throughput is the sum of the throughputs of all the sub-channels, this means that the data rate per sub-channel is only a fraction of the data rate of a conventional single carrier system having the same throughput. This property allows systems to achieve high data rates, while still maintaining symbol durations much longer than the memory of the channel. The

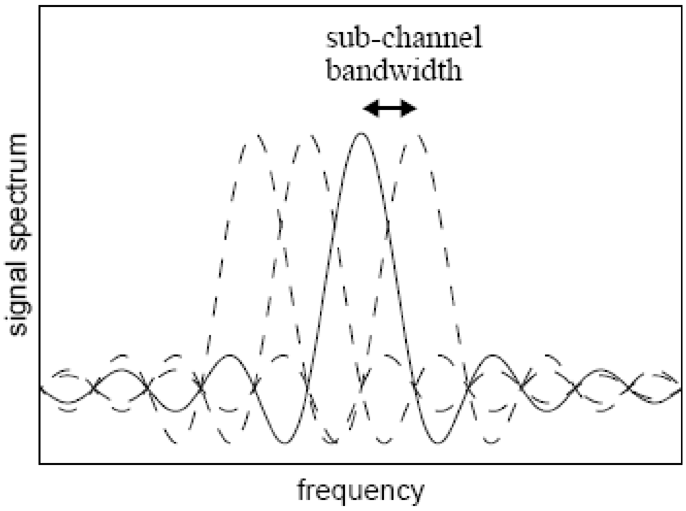


Figure 2.1: Spectrum of OFDM sub-carriers [9].

result of this parallel modulation technique is that we avoid complex channel equalization and it reduces the susceptibility to various forms of impulse noise [8].

OFDM can be implemented by using the discrete fourier transform (DFT). From signals and systems theory we know that the sinusoids of the DFT form an orthogonal basis set, and a signal in the vector space of the DFT can be represented as a linear combination of the orthogonal sinusoids. One way to interpret the DFT is that the transform correlates its input signal with each of the sinusoidal basis functions. If there is some energy at a certain frequency, the correlation of the input signal and the basis sinusoids will show a peak at the corresponding frequency. In the OFDM transmitter this transform is used to map an input signal onto a set of orthogonal sub-carriers. At the receiver the transform is used again to process the received sub-carriers. By combining the signals from the sub-carriers one can estimate the source signal from the transmitter [6].

In practice, OFDM systems are implemented using a combination of fast fourier transform (FFT) and inverse fast fourier transform (IFFT). These are the equivalent versions of the DFT and IDFT, respectively. The reason for this is that FFT are much more efficient to implement. The OFDM transmitter handles the source symbols (e. g. QPSK) in the frequency-domain. An IFFT block uses these symbols as input and transform the signal into the time-domain. The IFFT block takes N symbols, where N is the number of sub-carriers, and maps each symbol onto a sinusoidal basis function. Since

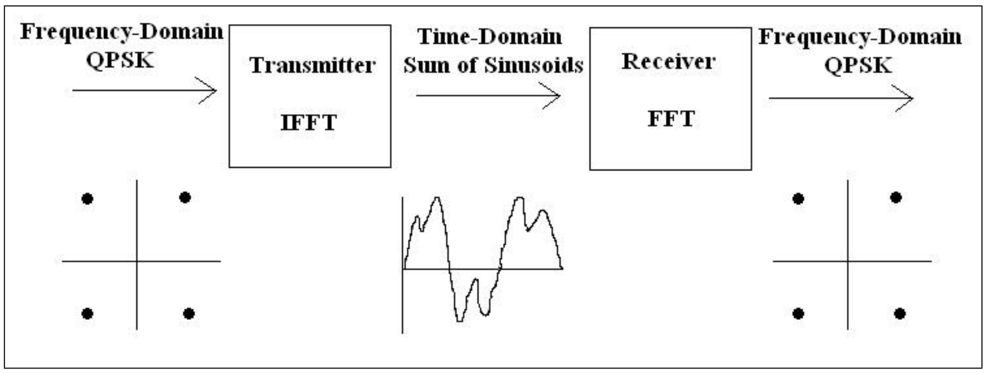


Figure 2.2: A simple OFDM system [6].

the input symbols are complex, the value of the symbol determines both amplitude and phase of the sinusoid. The output of the IFFT block is a OFDM symbol which are the summation of the N sinusoids. The length of the OFDM symbol is NT, where T is the period of the input symbol. At the receiver the signal is transformed into frequency-domain again by an FFT block and the output is the symbols that where sent to the IFFT at the transmitter [6]. In figure 2.2 a block diagram of a simple OFDM system is illustrated.

The presence of a multipath channel is a major problem for most wireless

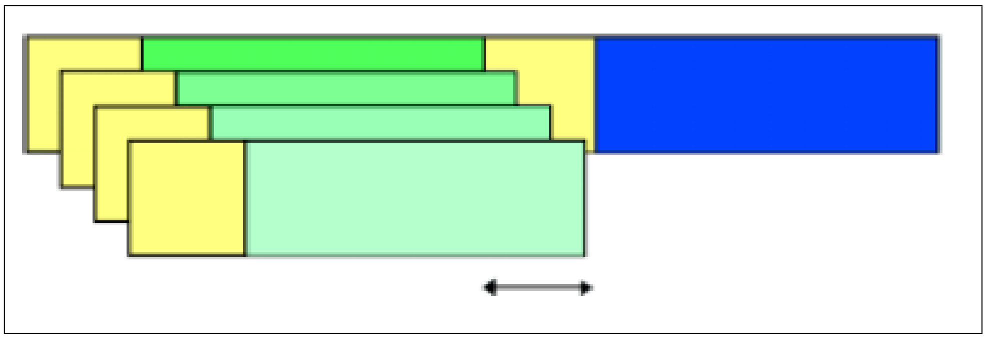


Figure 2.3: Intersymbol interference: the green symbol was transmitted first, followed by the blue symbol [6].

systems. In a multipath channel the signal reflects of several objects. At the receiver this results in multiple delayed versions of the transmitted signal, which causes the signal to be distorted. For an OFDM system this will cause two problems, intersymbol interference (ISI) and intrasymbol/intercarrier interference (ICI). Intersymbol interference occurs when the received OFDM symbol is distorted by the previous OFDM symbol. This problem can be solved by using a guard interval. By choosing an appropriate number of sub-carriers in the system one can make the length of the OFDM symbol longer than the time span of the channel. This way the intersymbol interference will only affect the first samples of the received OFDM symbol. If we now use a guard interval in front of each of the OFDM symbols we can remove the effect of intersymbol interference. The guard interval could for example be a section of all zeros. The length of the guard interval should be chosen such that it is longer than the time span of the channel. At the receiver the guard interval is discarded since it contains no useful information, and this way the intersymbol interference is discarded as well [6]. In figure 2.3 it is illustrated how the use of guard interval removes intersymbol interference.

Intrasymbol/intercarrier interference is unique to multi carrier systems. This interference is due to interference between a given OFDM symbol's own sub-carriers, which means that the OFDM symbol interferes with itself. The solution to this problems involves taking advantage of a discrete-time property. In discrete-time a convolution is only equivalent to a multiplication in the frequency domain if the signals are of infinite length or if at least one of the signals is periodic over the range of the convolution. By using a guard interval consisting of a given number of the last samples of the OFDM symbol, the OFDM symbol appears periodic when convolved with the channel and therefore the convolution is equivalent to a multiplication in the frequency domain. An important result of this is that the effect of the channel becomes multiplicative. This means that each sub-carrier's symbol will be multiplied by a complex number equal to the channel's frequency response at that sub-carrier's frequency. This multiplication will cause a complex gain (amplitude and phase) to each of the sub-carriers. In order to remove these effects a frequency domain equalizer is used. For a simple case with no noise this equalizer's response would be the inverse of the channel frequency response. A frequency domain equalizer is much more simpler than a time domain equalizer. At the receiver the cyclic guard interval is discarded, and this way it also removes the effects of intersymbol interference [6].

The decision of the various OFDM parameters is a tradeoff between various requirements. Often there are three main requirements to start with, bandwidth, bit rate and delay spread. By delay spread we mean the length of the channel impulse response. The guard time is dictated directly by the delay spread. The rule is that the guard time should be about two to four times the root-mean-squared of the delay spread, depending on the type of coding and QAM (quadrature amplitude modulation) modulation. Higher order QAM is more sensitive to ICI and ISI than QPSK (see section 2.1.2), while heavier coding reduces the sensitivity to such interference. The symbol duration should be much larger than the guard time in order to minimize the signal-to-noise (SNR) loss caused by the guard time. A practical choice is to make the symbol duration at least five times the guard time, this implies a 1-dB SNR loss. The number of sub-carriers is found by the bandwidth divided by the sub-carrier spacing, which is the inverse of the symbol duration. This number can also be determined by the required bit rate divided by the bit rate per sub-carrier. The bit rate per sub-carrier is defined by the modulation type, coding rate and symbol rate [7].

2.1.2 QPSK

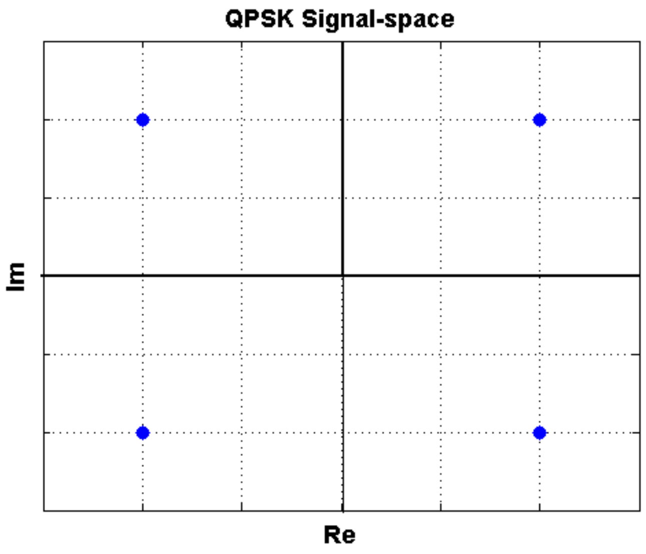


Figure 2.4: An example of a QPSK signal-space

In quadriphase-shift keying (QPSK) the information is carried in the phase of the transmitted signal. The phase of the carrier takes on one of four equally spaced values. These values could be $\pi/4$, $3\pi/4$, $5\pi/4$, and $7\pi/4$ as shown in figure 2.4. The transmitted signal may be defined as

$$y = \begin{cases} \sqrt{\frac{2E}{T}} cos\left[2\pi f_c t + (2i-1)\frac{\pi}{4}\right], & 0 \leq t \leq T \\ 0, & \text{elsewhere} \end{cases} \tag{2.1}$$

where i = 1, 2 ,3 ,4; E is the transmitted signal energy per symbol, T is the symbol duration and f_c is the carrier frequency [3]. Each possible value of the phase corresponds to a unique dibit, for instance as shown in table 2.1.

The probability of symbol error for QPSK on an additive white gausian noise

Table 2.1: Signal-space characterization of QPSK

Dibit	Phase
11	$\pi/4$
01	$3\pi/4$
00	$5\pi/4$
10	$7\pi/4$

channel (AWGN) may be written as

$$P_e \simeq erfc\left(\sqrt{\frac{E_b}{N_0}}\right), \tag{2.2}$$

where erfc() is the complementary error function, E_b is the signal energy per bit and $N_0/2$ is the noise power spectral density [3]. By using Gray encoding for the incoming symbols we can find the bit error rate of QPSK to be exactly

$$BER = \frac{1}{2}erfc\left(\sqrt{\frac{E_b}{N_0}}\right). \tag{2.3}$$

This means that a QPSK system achieves the same average probability of bit error as a binary PSK system but the QPSK system uses only half the channel bandwidth. Therefore for the same average probability of bit error, a QPSK system transmits information at twice the bit rate of a binary PSK system for the same channel bandwidth [3].

2.1.3 Differential-QPSK

Differential-QPSK may be viewed upon as the none coherent version of QPSK. By using differential coding one eliminates the need for a coherent reference signal at the receiver by combining two basic operations at the transmitter, differential encoding of the input binary data and QPSK. This means that the information do not lie in the received symbols, but in the phase difference between two succeeding symbols. This way the receiver will be independent of the phase of the carrier wave. A premise for this is that the phase of the carrier wave varies slowly, which means slow enough for the phase to be considered constant over two symbol intervals [3].

2.2 Environmental conditions and physical properties

The physical properties and the environmental conditions of the sea and the sea bottom have a great influence on acoustic waves, it is therefore important to have knowledge and understanding of these. In this section we will look at some of the properties and conditions that affects the underwater communications the most. Most of the material and figures in this section are collected from [4].

2.2.1 Geometrical spreading and absorption

Geometrical spreading is the spreading of energy in space. What we mean by this is that when a certain amount of energy is radiated from a source it spreads over a certain volume. This spreading is very much dependent on which kind of radiation we have. For the 1D case there will be no geometrical spreading because the volume we radiate into is just a thin line. When we have 2D radiation the energy spreads over a cylindrical surface. The amplitude of the wave will then decrease proportionally to the distance, $1/\sqrt{r}$. For the 3D case, which is the most relevant for real situations, the energy spreads in all directions. This means that the energy will spread across a spherical surface, and the amplitude of the wave decreases proportionally to the distance, $1/r$. The geometrical spreading for the different cases mentioned above is derived mathematically in [4].

When waves propagate there will be energy losses because some of the waves

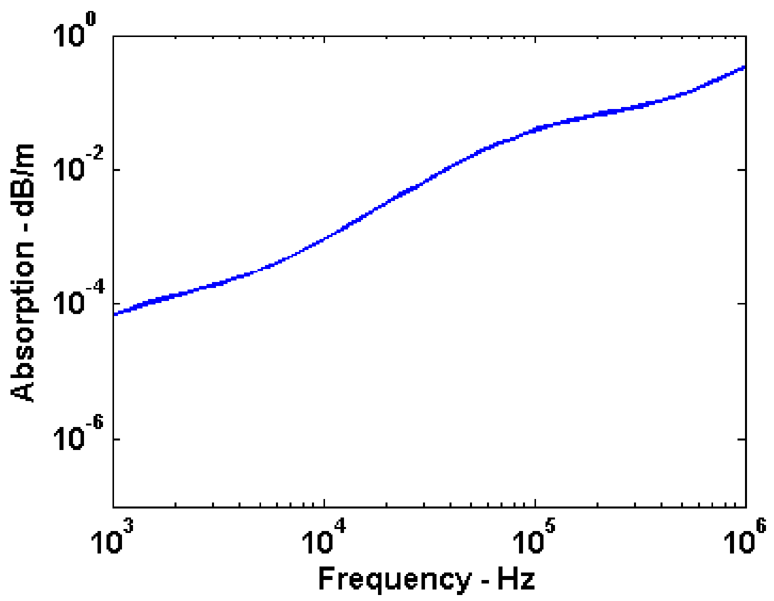


Figure 2.5: The attenuation in seawater as a function of frequency

energy will be converted to heat. This heat is created by friction or viscosity caused by the wave action. This absorption is proportional to the frequency squared. In figure 2.5 the absorption is plottet as a function of frequency for a wave propagating in seawater. As one can see this causes a frequency selective fading channel. For more details on this subject the reader can view [1].

2.2.2 Sound speed profile

The sound speed in the sea is dependent on ambient conditions, especially the water temperature, salinity and the surrounding pressure or depth of the water. A simple formula that is sufficiently accurate has been developed

$$c = 1448.6 + 4.618T - 0.0523T^2 - 1.25(S - 35) + 0.017D, \tag{2.4}$$

where c is the speed of sound, T is the temperature, S is the salinity (pro mille) and D is the depth in meters [4]. Typical sound speed profiles for

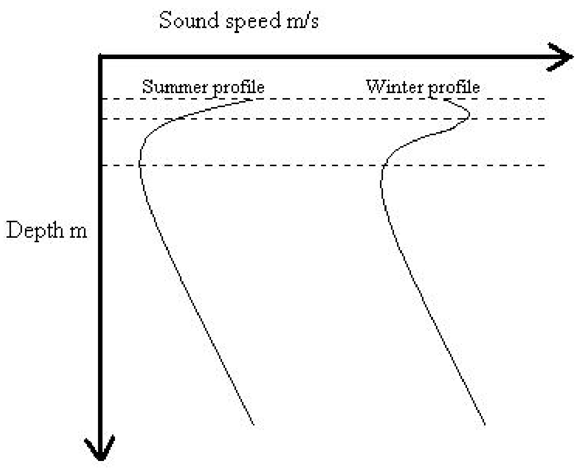


Figure 2.6: Typical sound speed profiles for summer and winter.

different seasons can be seen in figure 2.6. The upper layer near the surface is influenced by daily or seasonal changes in temperature. These changes is caused by heating and cooling, and as a result of ocean waves which mixes the water masses. The next layer is less affected by the surface conditions. Here the temperature decreases with depth and as a result the sound speed decreases. At great water depts, as shown in the lower layer in figure 2.6, the sound speed will increase though the temperature remains close to constant. This increase in sound speed is due to the increasing pressure at greater depths. This layer will appear at different depths depending on the location. In southern waters this depth will be at approximately 1000 m, while in northern waters it will be higher up.

The consequence of this depth dependent sound speed is that the sound will follow a curved path instead of a direct line. This impacts the propagation of acoustic waves in many ways. One result of this curved path is that there can be created acoustical shadow zones where little or no sound can reach directly from the source. In this case one is dependent on the reflections to transmit a signal. Another effect caused due to the depth dependent sound speed is the creation of a sound channel. This happens when there is a layer with a negative gradient above a layer with a positive gradient. If the source is placed in the middle of these to layers the sound will be concentrated along this depth. This sound channel results in cylindrical propagation and a transmission loss that is much smaller than the spherical transmission loss (2.2.1).

2.2.3 Doppler spread

When either the transmitter or/and the receiver in a communication system is in motion the signal will experience a frequency shift called a Doppler shift. The signal will also experience a Doppler shift when interfering with waves at the surface. The Doppler spread is a measure of the spectral broadening caused by these Doppler shifts in the communication channel. This Doppler spread is highly dependent on weather conditions such as wind.

2.2.4 Bottom interactions

In underwater acoustic the reflections from the bottom play a major role. Because of this it is important to have knowledge of how the bottom affects acoustic waves. If the bottom is infinitely hard all of the energy in the wave will be reflected, and there will be no loss. This is rarely the case. The upper layer of the bottom is often made up by sediments that are loose and less consolidated. These sediments can be silt, clay, sand or a mixture. The sound speed in this kind of sediments is usually a little higher than the sound speed in water. Since the sediments is not infinitely hard all the energy in the wave will not be reflected. Instead there will be a pressure wave transmitted into the bottom. Though the sediments are very loose there will still be a shear rigidity. That means that also shear waves are able to propagate in the material and will be transmitted into the bottom. The two transmitted waves (pressure and shear) will take the energy from the original wave making it weaker.

When we look at the case where there is a sediment layer on top of infinitely thick solid, the attenuation of the reflected waves can be calculated by using the reflection coefficient found in [4]:

$$R = \frac{r_{01} + r_{12} exp\left(2i\gamma_{p1} d_1\right)}{1 + r_{01} r_{12} exp\left(2i\gamma_{p1} d_1\right)}. \tag{2.5}$$

Here d_1 is the thickness of the sediment layer. r_{01} which is the reflection coefficient between the water and sediment layer, is given by

$$r_{01} = \frac{Z_{p1} - Z_{p0}}{Z_{p1} + Z_{p0}}, \tag{2.6}$$

where Z_{p1} and Z_{p0} is the specific impedance of the sediment layer and water respectively. The impedance of the medium is given by the product of the sound speed and the density. r_{12} is the reflection coefficient between the sediment layer and the infinitely thick solid and is given by

$$r_{12} = \frac{Z_{p2}cos^2(2\theta_{s2}) + Z_{s2}sin^2(2\theta_{s2}) - Z_{p1}}{Z_{p2}cos^2(2\theta_{s2}) + Z_{s2}sin^2(2\theta_{s2}) + Z_{p1}}. \tag{2.7}$$

In equation 2.7 Z_{p2} and Z_{s2} is the impedance of the infinitely thick solid for the pressure wave and the shear wave, respectively. θ_{s2} is the gracing angle of the transmitted shear wave. γ_{p1} in equation 2.5 is the vertical wave number for the pressure wave in the sediments and is given as

$$\gamma_{p1} = \frac{\omega}{c_{p1}} sin\theta_{p1}, \tag{2.8}$$

where θ_{p1} and c_{p1} is the gracing angle and the sound speed of the incident pressure wave, respectively. The the value of the reflection coefficient given by equation 2.5 can range between $|0|$ (e.g. very loose sediments) and $|1|$ (e.g. very hard bottom surface), depending on how hard the bottom surface is. The amplitude of the wave is then multiplied by the reflection coefficient to obtain the correct amplitude after the interaction with the bottom.

Chapter 3

Communication system

In this chapter a system for underwater communications is presented. The system is based on the OFDM principle. First the system will be described in general, before a detailed description of a implementation in MATLAB will be given.

3.1 The system

In underwater communications one faces a lots of different challenges and problems that are unique to this environment. Electromagnetic waves can only travel a short distance in water. Because of this one have to use acoustic waves instead. These acoustic waves have a very slow propagation velocity compared to that of electromagnetic waves. The slow propagation velocity cases long channel impulse responses as relatively strong reflections may arrive some time after the first arrival. This greatly influence the received wave and may causes interference. For long range communications this could be especially challenging. The sound speed varies with depth (see section 2.2.2), which may cause the acoustic waves to bend upwards or downwards. If one wishes to transmit a signal underwater over a great distance one may depend on the reflections from the surface and/or the bottom to receive the signal. This makes multipaths a huge challenge for underwather communication systems. In order to avoid these problems and thereby avoid interference the choice of modulation technique was OFDM (see section 2.1.1). In OFDM the data are transmitted over several sub-carriers in parallel. By having a low bit rate per sub-carrier, while still providing a large data rate totally, it makes the OFDM symbols much longer than the channel impulse response. This way the intersymbol interference (ISI) is reduced considerably. By also implementing guard interval the ISI is reduced even more. Another advantage of OFDM is that the flatness perceived by the narrow band channel of the sub-carriers also overcomes the frequency selective fading (see section 2.2.1) of the underwater acoustic channel.

Phase shift keying was chosen as the coding technique. This coding technique is simple and produces a constant amplitude signal, which reduces problems with amplitude fluctuations due to fading. By choosing quadrature phase shift keying (see section 2.1.2 on QPSK) one can achieve a higher data rate by sending two bits on every sub-carrier, but with a slightly higher bit error rate (BER). In order to keep the complexity of the receiver to a minimum it was finally decided to use differential-QPSK (see section 2.1.3). This way one avoids complex carrier tracking at the receiver because the data lies in the phase difference, and not in the received symbol itself. By choosing differential-QPSK one also solves a couple of problems in underwater communications. As mentioned in section 2.2.3 about Doppler spread, the movement of either the transmitter and/or receiver during transmission will cause a frequency shift. This will also occur when the acoustic wave interacts with surface waves. If the frequency shift is varying slowly, i.e. slow enough for the frequency to be considered constant over two symbol periods, the phase difference should remain unaffected and the data can be retrieved.

3.1.1 The transmitter

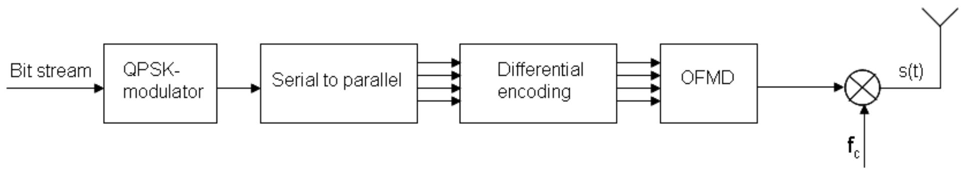


Figure 3.1: Block diagram of the transmitter.

In the figure above (3.1) a simple block diagram of the transmitter is illustrated. The bit stream is coded by a QPSK-modulator, which means that two by two bits are given a specified QPSK symbol. Then the data is converted from serial to parallel. This conversion is necessary because the input of the OFDM block needs to be parallel in order to map the symbols onto the sub-carriers. After the serial to parallel conversion comes the differential encoder. It is important that the differential coding is done on the parallel data, otherwise the differential coding will fail. This is because the frequency shift that differential coding should prevent affects each sub-carrier differently, and it will only work if the phase difference is found from two succeeding symbols from the same sub-carrier. In the differential coding block a reference signal is also created. This will be the first symbol to be sent and will be used by the receiver for synchronization. The OFDM block consists of three operations, zero padding, an inverse fourier transform and the insertion of guard interval. The zero padding is done to achieve oversampling and to center the spectrum of the data. This is done by simply inserting zeros at the middle of the parallel data. The inverse fourier transform, which in reality is a IFFT (see section 2.1.1 for details), takes the zero padded data as input. What the IFFT actually does is that it maps the data onto the sub-carriers and creates a time domain symbol which is the sum of all the sub-carriers. This operation also converts the data from parallel to serial again. Now a number of zeros are inserted in front of each OFDM symbol as a guard interval. In order to transmit the signal it is upshifted from baseband to the carrier frequency, f_c, before it is transmitted over the communication channel.

3.1.2 The receiver

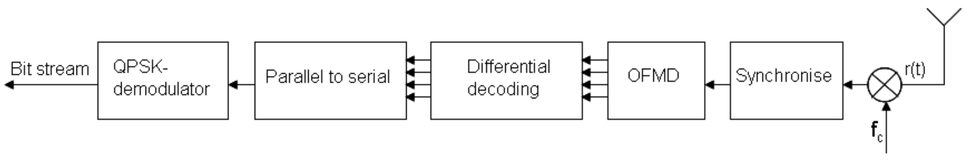


Figure 3.2: Block diagram of the receiver.

A simple block diagram of the receiver is shown in figure 3.2. The received signal is first downshifted from the carrier frequency, f_c, to baseband. In order for the OFDM block to use the correct window for the fourier transform the signal has to be synchronized. The synchronise block is essentially a correlation function, which correlates the received signal with the known reference symbol. By locating the maximum of the correlation function one is able to calculate the right window for the fourier transform. The OFDM block at the receiver is also made up of three operations. First the guard interval is discarded, then the received signal is transformed back into the frequency domain by an FFT (see section 2.1.1 for details). The zeros which were inserted at the middle of the data at the transmitter is then removed. Since we now have the data in the phase difference of the received signal, the next step is to decode the differential coding. The differential decoder finds the phase difference between two succeeding symbols from the same sub-carrier and the output is QPSK symbols. The parallel data consisting of QPSK symbols are first converted into serial data before it is demodulated. This is done by the QPSK-demodulator which demodulates a QPSK symbol into unique dibits. The result of this is a stream of bits out of the receiver which, depending on the communication channel, should be equal to the bit stream into the transmitter.

3.2 Implementation

In this section there will be given a detailed description of an implementation of the system described above. The implementation is done in the mathematical programming language MATLAB. The program is divided into a transmitter and a receiver part. The source code for the program is found in appendix C.

3.2.1 The transmitter

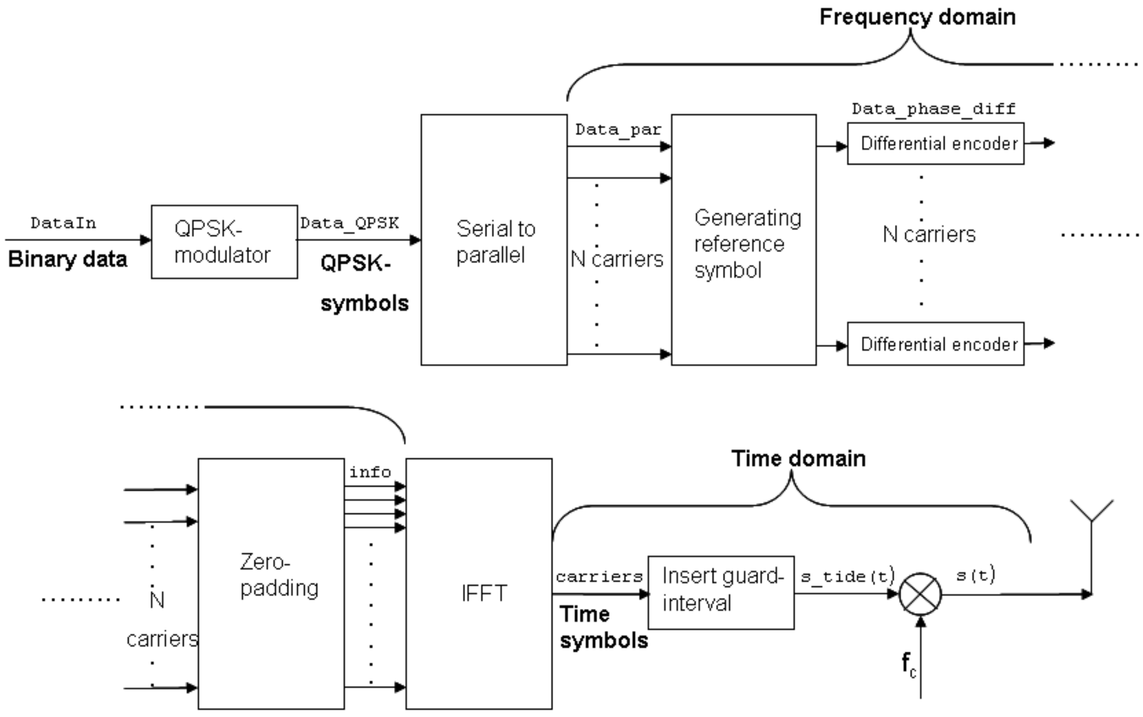


Figure 3.3: An overview of the implementation of the transmitter.

The figure above shows a detailed diagram of the implementation of the transmitter. Here the text just above the arrow indicate the name of the data vector or matrix at the given stage of the program. The text in bold just under the arrows indicate in what kind of form the data is.

The very first thing done in the transmitter implementation is to declare variables. In order to do this the program needs a few inputs. These inputs are the number of carriers N, the symbol length T_u, the number of OFDM symbols to be created M, the guard interval "*guard*" in seconds and the type of data to be transmitted. The type of data in this program can be random or it can be a sound recording. By using these inputs, the bandwidth B given by the transducer and sampling frequency F_s, the rest of the variables are calculated.

Now that the variables has been declared, the next step is to create the $DataIn$-vector. Depending on the type of data chosen, this vector is created randomly by using the function $rand()$ in MATLAB or from loading a specified sound file into MATLAB using the function $loadSound()$ (for details see appendix C). Both these methods results in a vector consisting of 1's and 0's with a length of $2 \cdot N \cdot M$. This binary vector is next modulated into QPSK by the function $QPSKmodulator()$. This function assigns two by two bits to a given QPSK-symbol (see figure 2.4) as shown in table 3.1, and the result is the vector $Data\_QPSK$ of length $N \cdot M$.

Table 3.1: QPSK-symbol mapping

Dibit	QPSK-Symbol
11	1+i
01	-1+i
00	-1-i
10	1-i

In order to prepare the data for differential coding and OFDM modulation the $Data\_QPSK$-vector is transformed from serial to parallel data. This is done by a simple for-loop and the $Data\_par$-matrix is created. This matrix has a number of columns and rows given by the number of OFDM symbols M, and the number of carriers N, respectively. As shown in figure 3.3 the next step is to generate a reference symbol. This reference symbol is a column of N rows consisting of QPSK-symbols. It will act as a staring value for the differential coding and will be known at the receiver. This reference symbol will also be used in the synchronization at the receiver. In reality this means that there will be transmitted $M+1$ symbols, but since the information will be contained in the phase difference only M effective data symbols will be transmitted. The differential encoding is done by mapping the QPSK-symbols as a complex phase as shown in table 3.2.

Table 3.2: Phase shift mapping

QPSK-Symbol	Phase shift
1+i	-1
-1+i	i
-1-i	1
1-i	-i

The reference column is then multiplied by the first data column, now mapped as a phase, and the result is a new column. Now the information lies in the phase difference between the new column and the reference column. Next the new column is multiplied with the second phase column and another column is created and so on. This means that after this operation we have a matrix, called *Data_phase_diff* (see figure 3.3), with $M+1$ colums and N rows where the *Data_par*-vector can be found by calculating the phase difference of the columns. This operation is also illustrated in figure 3.4.

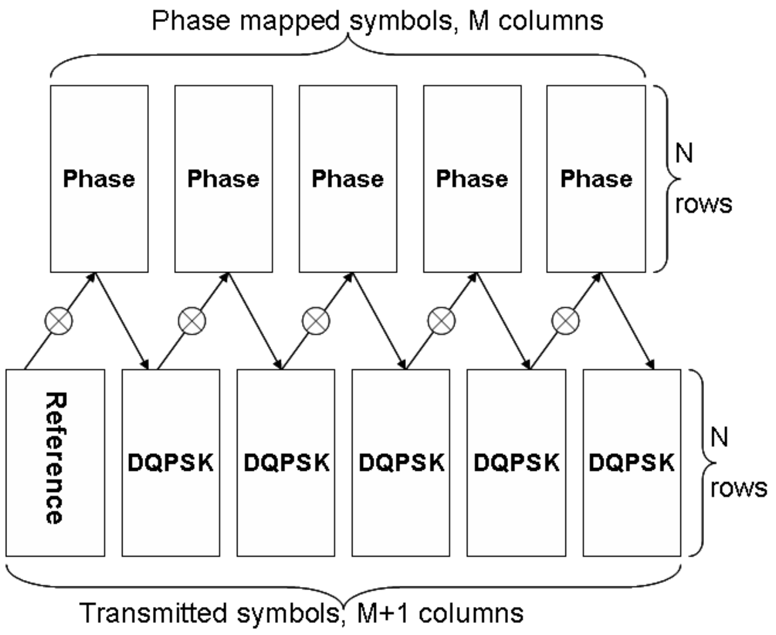


Figure 3.4: Illustration of differential encoding.

After the differential encoding the OFDM operation is implemented. The first step is zero padding. This is simply done by placing a number of zeros, depending on the sampling frequency f_s, in the middle of the rows of the *Data_phase_diff* matrix. This will center the spectrum of the data and provide oversampling. The second step of the OFDM operation is done on the columns of the $info$ matrix one by one using a for-loop. In this operation the columns are transformed into OFDM symbols by using the MATLAB function $ifft()$. Up to now the data has been handled in the frequency domain, but now the $ifft()$ transforms the data into the time domain. The time domain signal *carriers* is the sum of all the non-zero sub-carriers of the $info$ data (see figure 3.3). In the same loop as the second OFDM operation is preformed, the guard interval is inserted and the signal is also shifted up in frequency. Inserting the guard interval is done by adding a number of zeros, specified by the input ”*guard*”, in front of the OFDM symbols (except for the reference symbol) while the shift in frequency is done simply by multiplying the *carrier* signal by the exponential function, $exp(i \cdot 2 \cdot \pi \cdot f_c \cdot t)$. This upshifts the spectrum of the signal to the predefined carrier frequency, f_c. The OFDM symbols and their guard intervals are then added together creating the total signal $s(t)$, which now are ready to be transmitted on the communication channel. In order to get the signal to a format which easily can be transmitted it is stored as a wave-file using the MATLAB function *wavwrite*().

3.2.2 The receiver

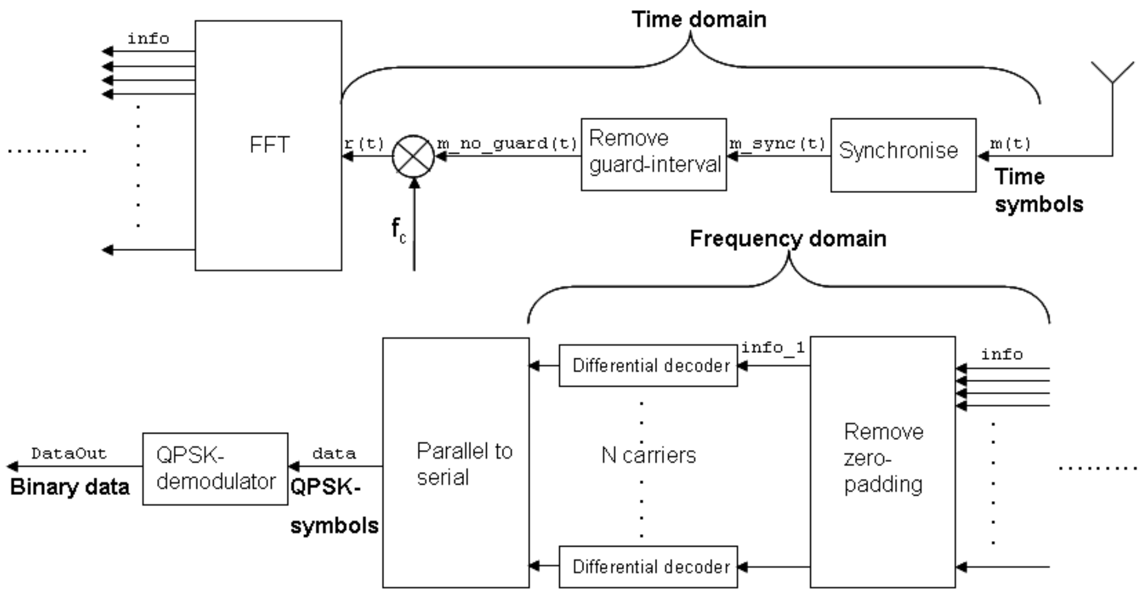


Figure 3.5: Block diagram of the receiver.

Figure 3.5 shows a detailed description of the implementation of the receiver in MATLAB. As in figure 3.3 the text just above the arrows is the name of the data vector or matrix at the given stage of the program, while the bold text just below indicate in what state the data are at.

The received signal $m(t)$ from the communication channel, which is recorded as a wave-file, are first read into the program using the MATLAB function *wavread*(). Now the signal needs to be synchronized in order for the FFT-window at the OFDM stage to be correct. This is done by correlating the reference signal created at the transmitter with the received signal. In the program this operation is done by the MATLAB function *xcorr*() which returns the correlation function of the to signals. Then by using the MAT-

LAB function *max*() the location of the maximum value of the correlation can be found and the start of the received signal can be calculated. Now that the signal is synchronized the guard interval can easily be discarded. This is done symbol by symbol in a for-loop. In the same loop the signal is down-shifted, transformed into the frequency domain and the zero padding is removed. The down-shifting from the carrier frequency f_c to baseband is done by a multiplication with an exponential function, the same way as the up-shifting at the transmitter, but now with a minus sign in the exponential function, $exp(-i \cdot 2 \cdot \pi \cdot f_c \cdot t)$. Transforming the signal $r(t)$ (see figure 3.5) into frequency domain is done by the MATLAB function $fft()$. Now, depending on the communication channel, the output of the $fft()$ should be equal to the zero padded signal ,$info$, at the receiver (see figure 3.3). The zero padding is simply discarded by just removing all the sub-carriers with zero-value, resulting in the matrix, $info\_1$, which now consist of only the sub-carriers containing data.

The QPSK-symbols containing the data is now in the phase difference between two succeeding columns in the matrix $info\_1$. The task of the differential decoder is to find and decide this phase difference. By multiplying a symbol at a given sub-carrier by the complex conjugate of the previous symbol at the same sub-carrier the phase difference is found. This is shown in equation 3.1 below:

$$e^{i\theta_2} \cdot e^{-i\theta_1} = e^{(\theta_2 - \theta_1)} \tag{3.1}$$

In ideal situations the phase differences found in the differential decoder should be equal to the values shown to the right in table 3.2. Because this is seldom the case due to the communication channel the value of the phase differences has to be decided. This is done by first comparing the absolute values of the imaginary and the real real parts of the phase difference. If the imaginary part is the biggest it is most likely that the phase difference is either i or $-i$ or if the real part is the biggest it is most likely that the phase difference is either 1 or -1. The sign is then decided by comparing the value to zero. When the value of the phase difference has been decided it is decoded into QPSK symbols according to table 3.2. The differential decoded matrix is then converted into a vector (parallel to serial) and the QPSK symbols are demodulated into dibits according to table 3.1. Both these operations are done by the function *QPSKdemodulator*().

The bit stream *DataOut* out of the receiver , should ideally be exactly the same as the bit stream *DataIn* in to the transmitter. This of course depends on the communication channel the received signal has been transmitted on. If the type of input data were "*sound*" the bit stream is transformed into a wave-file by the function *writeSound*() (for details see appendix C). In order to characterize the effect the channel has had on the transmitted signal the last thing implemented in the program is a error check routine which finds the number of both the QPSK-symbol errors and the bit errors. This is done by comparing the $Data\_QPSK$-vector (transmitter) with the $data$-vector (receiver) and the $DataIn$-vector with $DataOut$-vector. The bit error ratio is also calculated.

Chapter 4

Simulations

In this chapter the system that where described in chapter 3 will be tested and classified through simulations. The first simulation will act as an demonstration of the OFDM system. Then the system will be tested under different channel parameters. These parameters are noise, phase error and multi paths. FFT-offset will also be simulated.

4.1 OFDM simulation

Table 4.1: Numerical values for the OFDM parameters.

Paramenter	Value
Type of data	"*sound*"
Bandwidth B	2560 Hz
Number of carriers N	512
Symbol length T_u	200 ms
Sampling frequency F_s	192 kHz
Carrier frequency F_c	38 kHz
Elementary period $T = 1/F_s$	5.2e-6 s
Guard interval δ	50 ms
Number of symbols M	20

In this section we will demonstrate the program described in section 3.2. The program will simulate a system for underwater communication using OFDM under ideal conditions. This means that there will be no noise, phase error or multipaths influencing the transmission. The numerical values for the OFDM parameters used in this simulation are shown in table 4.1 and are calculated according to [7]. The bandwidth and carrier frequency are given by the transducer, while the sampling frequency are given by hardware used for the playback of the signal wave-file. The data type "*sound*" means that the binary data to be transmitted is a sound-file. As can be seen in table 4.1 there will be transmitted 20 OFDM symbols with 512 sub-carriers. Remembering that the QPSK modulation enables us to transmit 2 bits per sub-carrier, the total number of bits to be transmitted is 20480. The total symbol time including the guard time is 250 ms. The total transmission time, including the reference symbol, is then 5.2 seconds. This results in a bit rate for the system of approximately 4 kbits per second.

4.1.1 Transmission

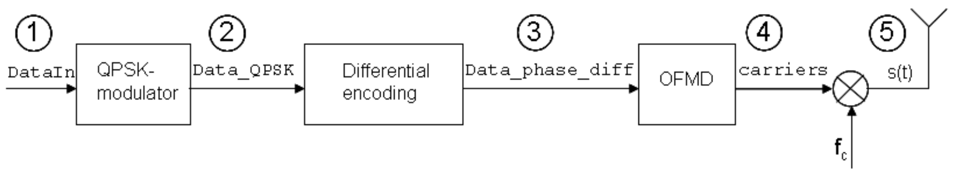


Figure 4.1: The transmitter.

The figure above gives a simple overview of the different stages of the transmitter in the program. The text above the arrows indicate the name of the variable containing the data to be transmitted.

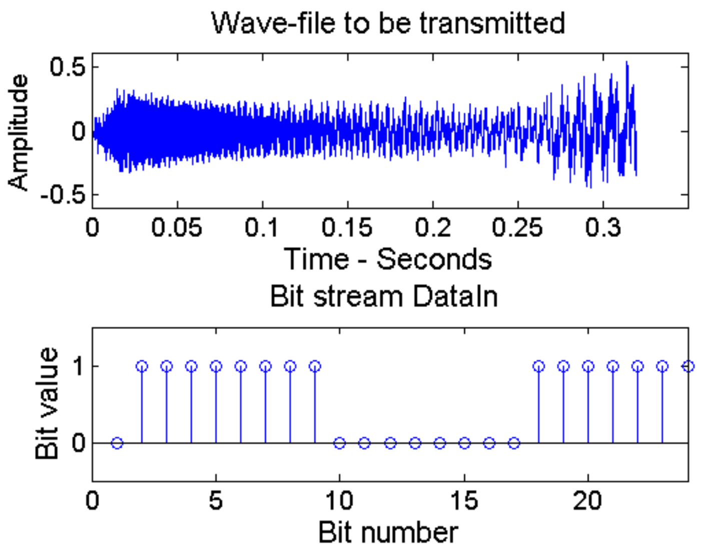


Figure 4.2: The wave-file and the bit stream *DataIn*.

In the upper part of figure 4.2 a plot of the wave-file which will be transmitted over the communication channel is shown. This is a mono wave-file sampled at 8000 Hz and with 8 bits resolution. The samples are converted back into binary values and inserted as serial data in the *DataIn*-vector (stage 1 in figure 4.1). A plot of the first 24 bits which makes up the 3 first samples of the wave-file are shown in the lower part of figure 4.2.

In stage 2 the bit stream *DataIn* is now modulated into QPSK. This means that two by two bits are grouped together to make up a complex QPSK symbol. The real and imaginary part of the first 12 symbols in the *Data_QPSK* variable are plotted to the left in figure 4.3 while the complex QPSK symbols are plotted to the right. By combining the real and imaginary values in the figure and by using table 3.1 the bits from figure 4.2 can be found.

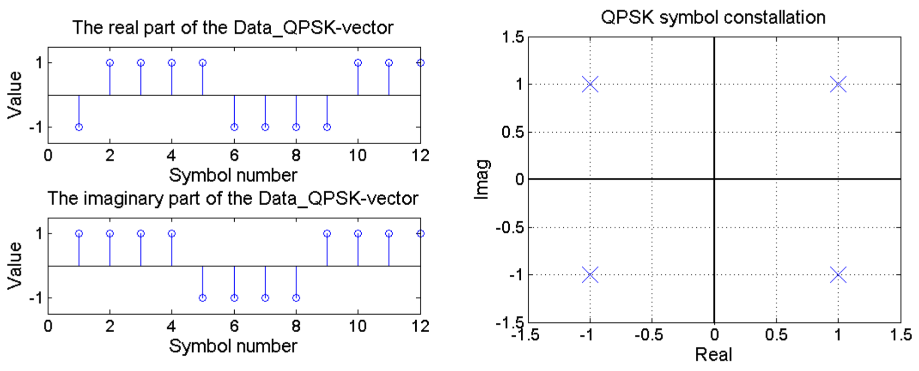


Figure 4.3: Plot of the first 12 QPSK symbols to be transmitted.

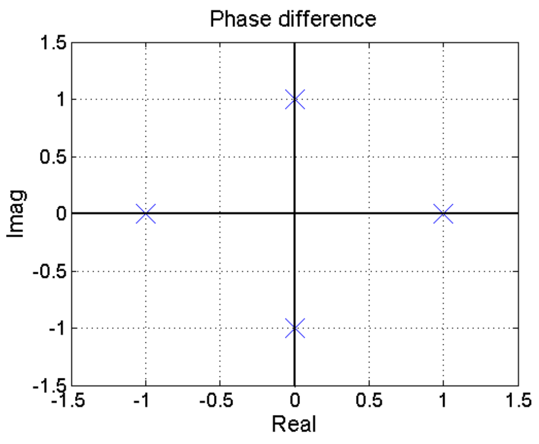


Figure 4.4: The mapped complex phase differences.

Because of the differential coding it is not the QPSK symbols shown in figure 4.3 that are transmitted over the channel. First the QPSK symbols are mapped as a phase difference according to table 3.2. This mapping is illustrated in figure 4.4. By multiplying this phase differences with a known reference QPSK symbol we get another set of QPSK symbols, *Data_phase_diff* (stage 3), but now the original QPSK symbols in figure 4.3 lies in the phase difference. It is the QPSK symbols in the *Data_phase_diff* variable that are transmitted over the communication channel.

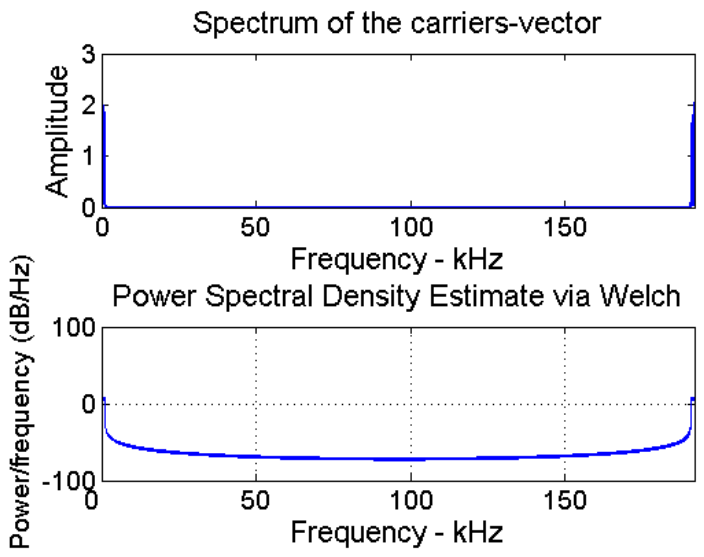


Figure 4.5: The frequency spectrum at the base band.

In order to achieve over-sampling and to center the spectrum the QPSK symbols are zero padded before they are mapped onto the sub-carriers. In the figure above (4.5) the result of this operation can be observed (stage 4). In this figure there is two plots of the frequency spectrum of the time domain base band signal, *carriers*. The first is a ordinary FFT plot while the second is a plot made by the MATLAB function *pwelch*. This function computes the power spectral density, not the power spectrum. In both plots we see that the energy is concentrated just above zero frequency and below the sampling frequency at 192 kHz. In reality this means that the energy is centered around zero frequency.

Figure 4.6 shows the frequency spectrum of the signal $s(t)$ after it has been shifted up to the carrier frequency (stage 5 on figure 4.1). Here we see that the spectrum is centered at 38 kHz and that the bandwidth is approximately 2.5 kHz. The resulting time domain signal is shown in figure 4.7. The 21 OFDM symbols, where the first is the reference symbol, is clearly visible be-

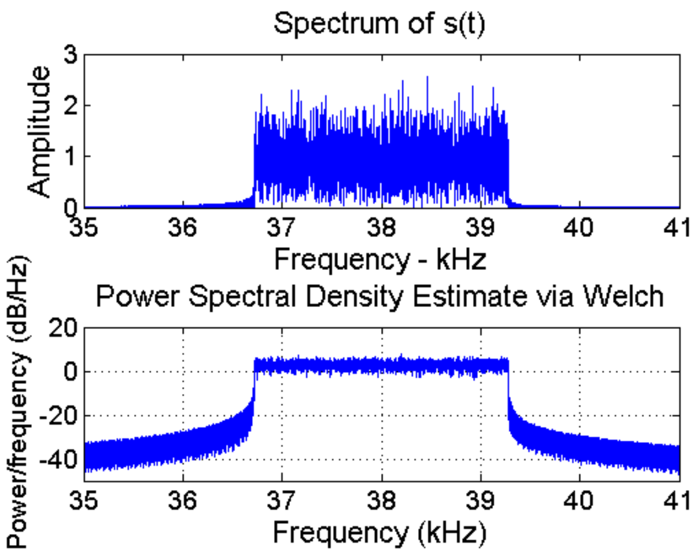


Figure 4.6: The frequency spectrum after the upshift to the carrier frequency.

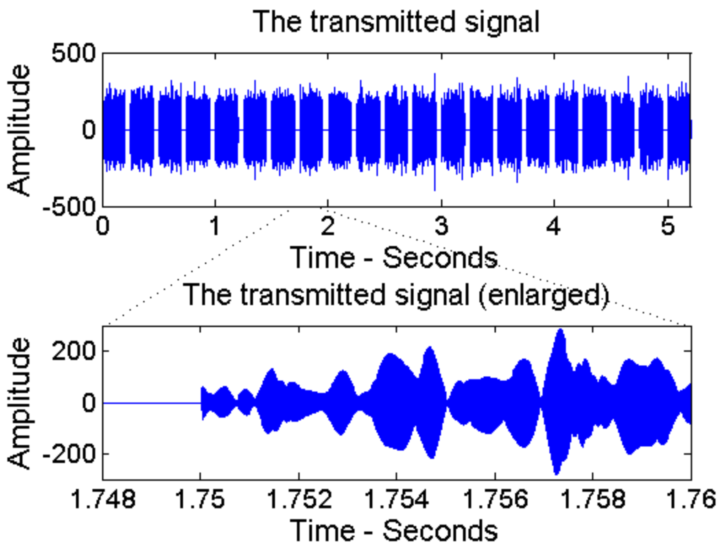


Figure 4.7: The signal to be transmitted over the channel.

cause of the inserted guard interval. The symbols may look like just random amplitudes at distant, but when enlarged the shape is more regular. The reason for this shape is because the data are mapped onto sub-carriers at different frequency. This means that the resulting signal is just a sum of these frequencies, and this creates is the shape shown in the lower part of figure 4.7.

4.1.2 Reception

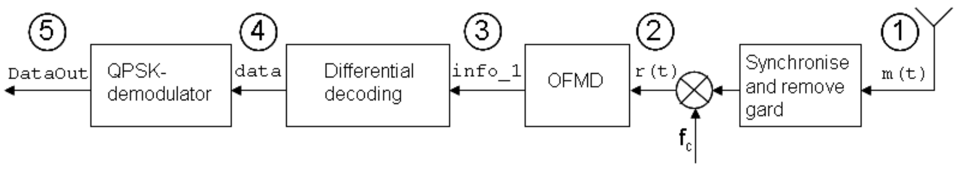


Figure 4.8: The receiver.

Figure 4.8 gives a simple overview of the different stages of the receiver in the program. The text above the arrows indicate the name of the variable containing the received data.

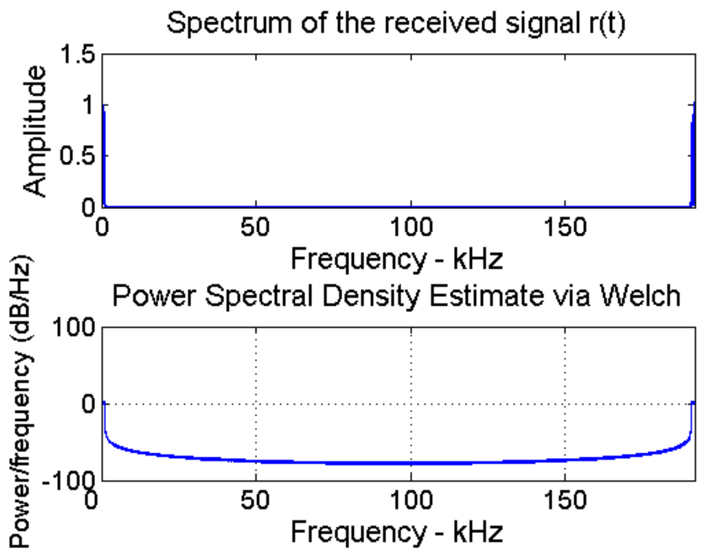


Figure 4.9: The frequency spectrum of the received signal after downshift to baseband.

Under ideal conditions the channel does not influence the transmitted signal, and the received signal at receiver (stage 1 in figure 4.8) is exactly the same as the transmitted signal (stage 5 at the transmitter) shown in figure 4.7. After the received signal has been synchronized and the guard interval has been removed, it is down shifted from the carrier frequency to base band frequency. The resulting frequency spectrum of the signal is plotted in figure 4.9 (stage 2). By comparing this spectrum to the spectrum in figure 4.5 we can see that these spectrums are identical.

In stage 3 the received signal $r(t)$ has been transformed into the frequency domain and the zero padding has been removed. The signal $info\_1$, is made up of QPSK symbols where the information lies in the phase difference. By multiplying the QPSK symbol with the previous received QPSK symbol the complex phase difference is found. In figure 4.10 the resulting complex phase differences are plotted and one can easily see that they do not differ much from the complex phase differences plotted in figure 4.4. If we were

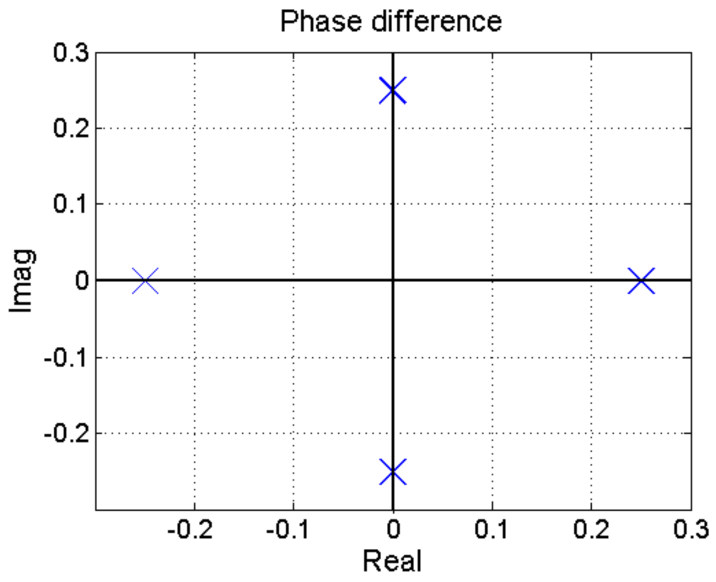


Figure 4.10: The complex phase differences at the receiver.

simulating with a realistic channel the values of the phase differences would differ quite a bit and could cause errors when deciding if the phase difference is i, -i, 1 or -1. Because we here simulate under ideal conditions the phase decision is obvious. After the decision the the original QPSK containing the data are found using table 3.2. The real and imaginary part of the first 12 QPSK symbols in the *data* variable are shown in figure 4.11.

The QPSK demodulator retrieves the two bits from each QPSK symbol according to table 3.1 resulting in the bit stream *DataOut*. The first 24 bits are shown in the upper part of figure 4.12. The bits in *DataOut* are then grouped together into groups of eight bits which together makes the samples of the transmitted wave-file. The result of the demodulation is zero bit errors and the lower part of figure 4.12 shows a plot of the resulting wave-file which

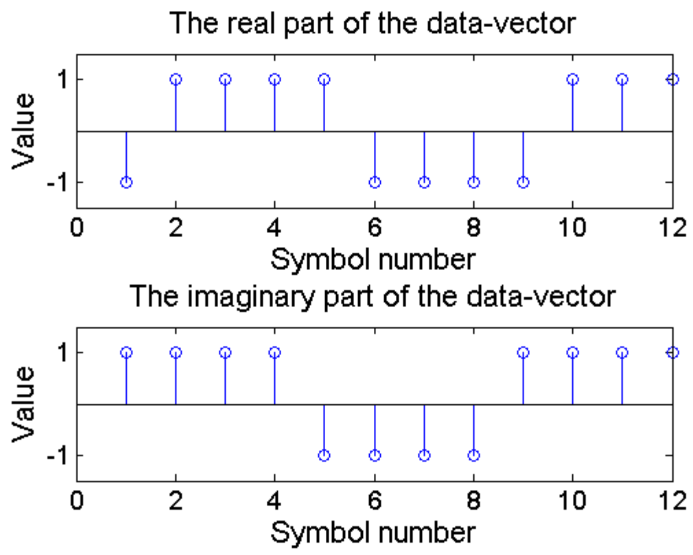


Figure 4.11: The received QPSK symbols at the receiver.

is identical to the transmitted wave-file plotted in figure 4.2.

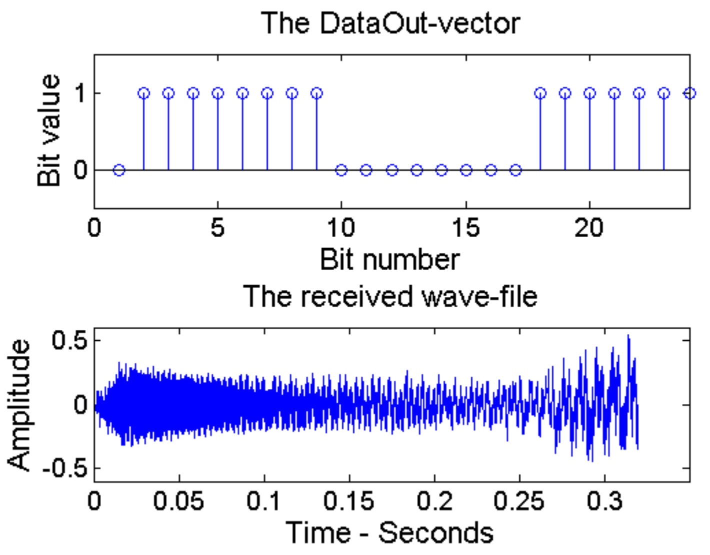


Figure 4.12: The first 24 received bits at the receiver and the resulting wave-file.

4.2 Channel simulations

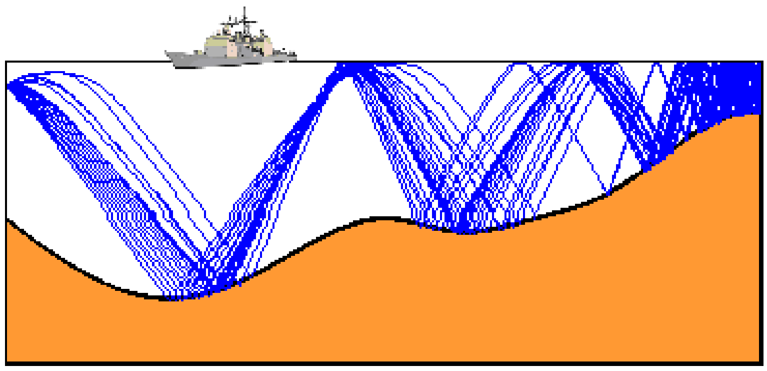


Figure 4.13: Illustration of an underwater channel

The conditions when transmitting data is seldom ideal. In a real communication channel there are many sources of errors which will affect the received signal. In the case of underwater communication some are more serious than others. There are many sources of noise in a underwater environment, this could be ship traffic, rain or other sources of sound which travels underwater. A noisy channel will cause a low signal to noise ratio and make it more challenging to retrieve the transmitted data. Also movement by either the transmitter or the receiver, together with waves and carrier offsets will lead to a phase error which also can cause errors at the receiver. Reverberations and multipaths are especially challenging in an underwater environment. The relatively low wave speed causes a long channel impulse response. All of the mentioned effects could also cause a offset in the synchronization at the receiver. In this sections we will simulate and illustrate how these effects affects the OFDM-system. The methods used in MATLAB for these simulations can be viewed in appendix C.

4.2.1 AWGN

In order to simulate noise in the channel, we add additive white gausian noise (AWGN) to the transmitted signal. This noise is uncorrelated and has a zero mean. This is done by simply adding the noise to the transmitted signal. The variance of the noise gives the level of the noise and it is calculated from a given signal to noise ratio.

Figure 4.14 illustrates the effect of the noise. Here the spectrum of the received and the transmitted signal are plotted and the signal to noise ratio of the channel has been set to 18 dB. The noise level can easily be seen to the left in the figure. The received and transmitted signal in the time domain

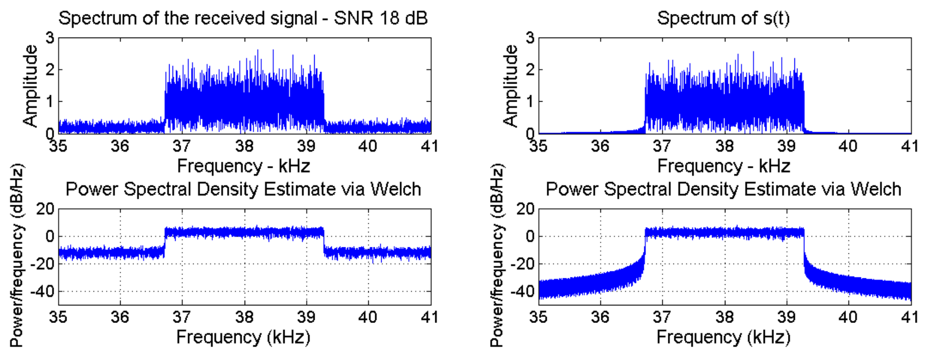


Figure 4.14: The received signal with and without noise.

can be seen in figure 4.15. Here we see that the received signal has lost much of its regular shape due to the noise. It is also more difficult to separate the guard interval from the OFDM symbol.

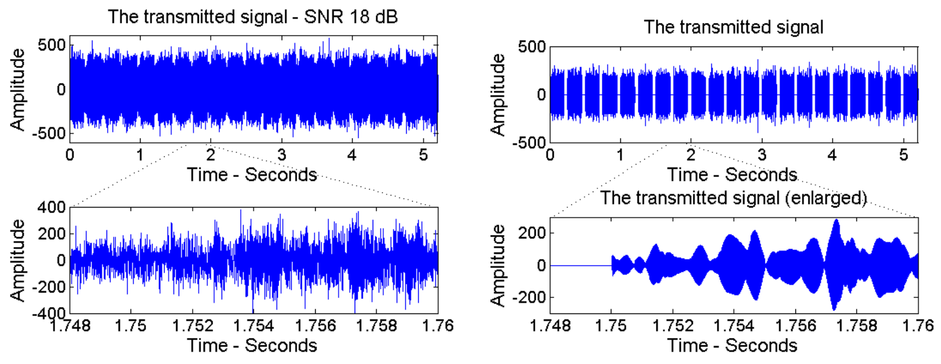


Figure 4.15: The received signal with and without noise.

The effect the noise has on the received signal is best illustrated by plotting the complex phase difference calculated in the differential decoder. This is shown in figure 4.16. Here we see the complex phase difference when the signal to noise ratio is inf, 30 dB, 25 dB and 18 dB. Inf means that the signal to noise ratio is infinity, i.e. there is no noise present. Here we see that the noise causes a greater variance in the calculated phase difference than it had without noise. This variance increases with the level of noise, and greatly increases the probability for a decision-error. Though the phase difference now has a higher variance it is still easy to see that it is centered around i, -i, 1 or -1 (multiplied by a amplitude factor depending on the transmission loss), and the decision operation should be error free. The resulting wave-file

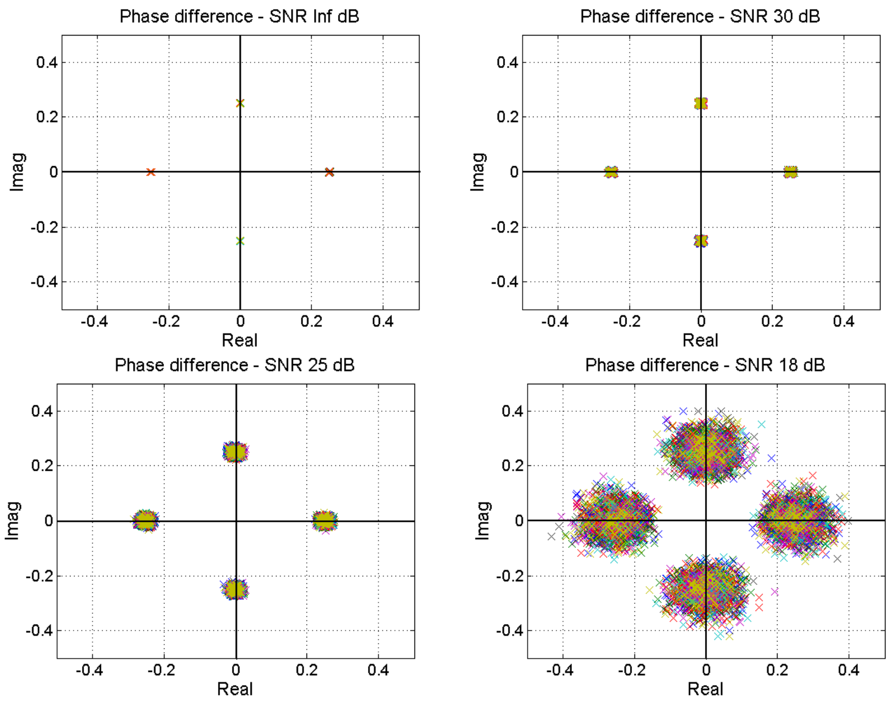


Figure 4.16: The complex phase difference with noise.

is plotted in figure 4.17. By comparing this plot to the plot in figure 4.2 we see that it is without any bit errors.

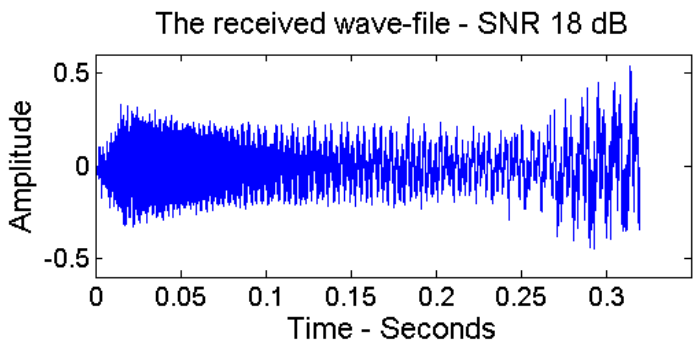


Figure 4.17: The received wave-file with a 18 dB SNR.

The bit error ratio (BER) is a very important measure for communications systems. This is because it tells us something about how a certain signal to noise ratio (SNR) affects the received data. In figure 4.18 the calculated bit error ratio for the OFDM system is plotted against the SNR when only

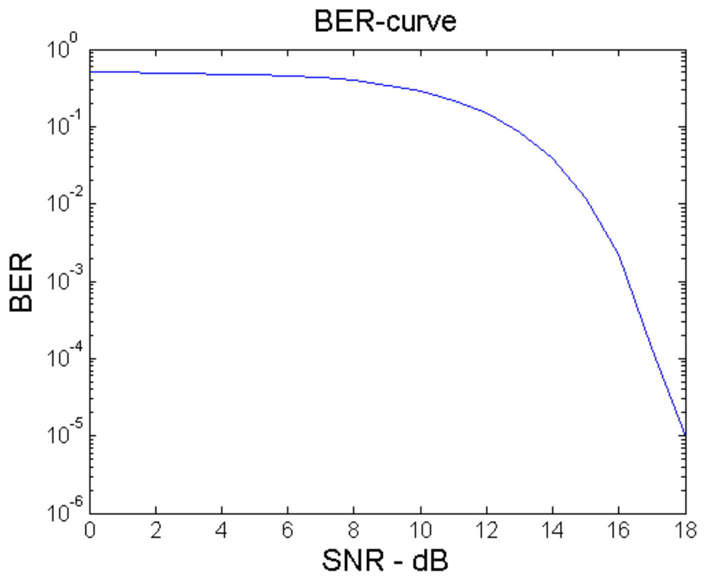


Figure 4.18: The bit error ratio as a function of SNR.

AWGN is present on the channel. Here we see that the error is quite severe at SNR's up to about 12 dB. From 12 dB and up the error drops rapidly and from 18 dB the the bit error is neglectable. This BER-curve tells us how much signal energy we need to use in order to achieve a certain probability for bit error. In this case a SNR above 17 dB should be sufficient.

4.2.2 Phase error

During the transmission time phase error can occur. This is the reason it is so important to either use carrier tracking at the receiver or differential coding. In this simulation this error is simulated by adding a linear increasing phase. This is done by multiplying the transmitted signal with the exponential function $exp(i\theta)$, where θ is the phase error plotted in figure 4.19. As we can see in the figure the phase error increases from zero to 2π during the transmission time of 5.2 seconds. This mean that at 2.6 seconds the phase of the transmitted signal has rotated 180 degrees from its original position and the phase error from one OFDM symbol to the next is 17 degrees (0.299 rad).

The effects of this phase error is not easy to see when looking at the received time domain signal, but becomes clear when the phase difference is calculated in the differential decoder. The resulting complex phase differences is plotted in figure 4.20. The most obvious observation is that the phase difference has been rotated approximately 15-20 degrees to the right, which

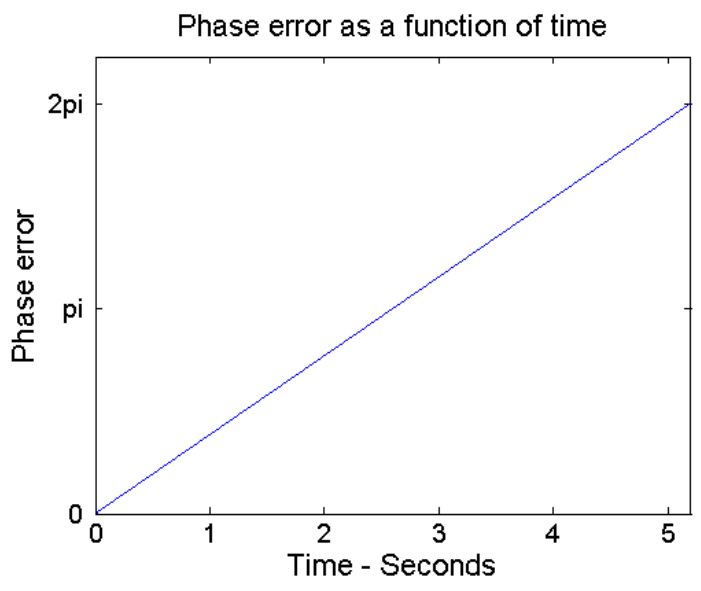


Figure 4.19: Phase error as a function of time.

is consistent with the phase error from one OFDM symbol to the next. We also see that the variance of the phase difference has increased some.

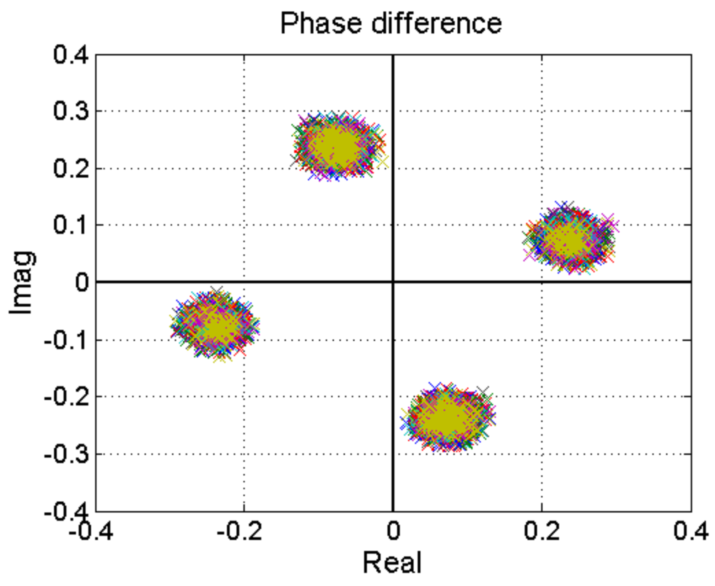


Figure 4.20: The complex phase difference with phase error.

Even though the phase differences has been rotated and the variance increased it should not cause a problem in this case. This is because the decision operation has no problem deciding if the phase are i, -i, 1 or -1 (multiplied by a amplitude factor depending on the transmission loss). As expected the resulting wave-file shown in figure 4.21 is without errors.

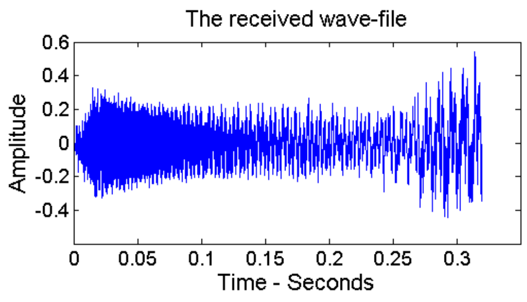


Figure 4.21: The received wave-file with phase error.

When phase error is present on the channel together with additive white gausian noise the probability for error increases. In figure 4.22 the bit error ratio for a channel with only noise present is compared to the bit error ratio when phase error also is present on the channel. Here we see that both the curves start out at a BER of approximately 0.5, but the difference becomes more clear when the SNR increase. In order to achieve a BER in the order of 10^{-4} the phase error increases the demand of the SNR with about 3 dB.

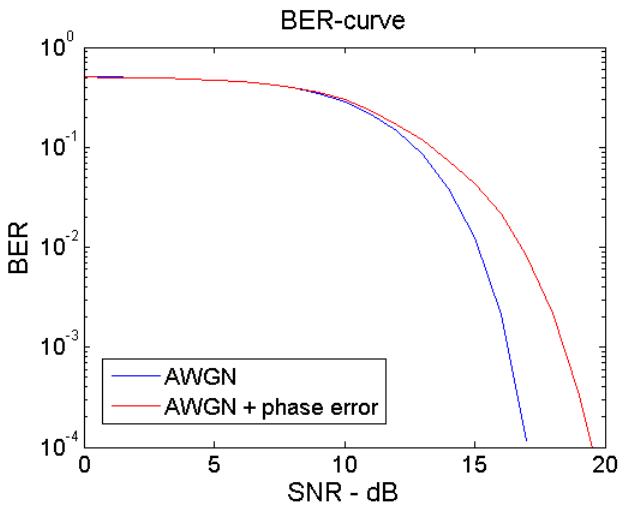


Figure 4.22: The bit error ratio as a function of SNR.

4.2.3 Multipaths

In underwater communications multipaths inflicts serious distortions to the received signal. This is due to the long channel impulse response caused by the low wave speed in the water.

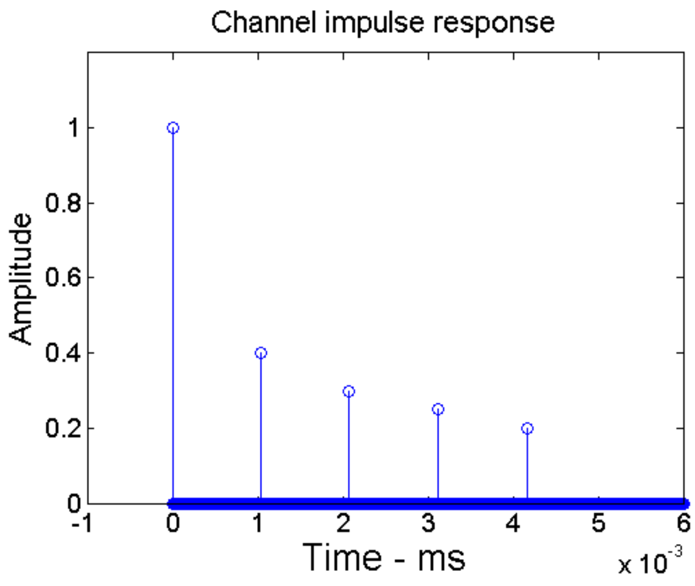


Figure 4.23: The channel impulse response.

Figure 4.23 shows a simple impulse response for a underwater channel. The peaks are arrivals at the receiver while the amplitude indicate how strong the arrival is. In order to simulate multipaths this impulse response is convolved with the transmitted signal. The resulting signal is then the sum of five transmitted signal where four have a delay and a weaker amplitude. In figure 4.24 the transmitted signal and the resulting received signal is plotted.
Here we see that the received signal has an echo of the same length as the impulse response. It is also clear that the shape of the received signal differs from the shape of the transmitted signal. This is caused by the summation of the five signals. Some of the main reasons for choosing OFDM lies in its abilities to handle multipaths. The symbol length is much longer than the length of the channel impulse response. This decreases the intersymbol interference (ISI) considerably. By inserting a guard interval in front of each OFMD symbol it is reduced even more. This can be seen in figure 4.24. At the transmitted signal the guard interval is clearly visible. But the guard interval at the received signal now contains echoes of the transmitted signal. This prevents the echoes causing distortions at the beginning of the next OFDM symbol. The guard interval is then just discarded along with

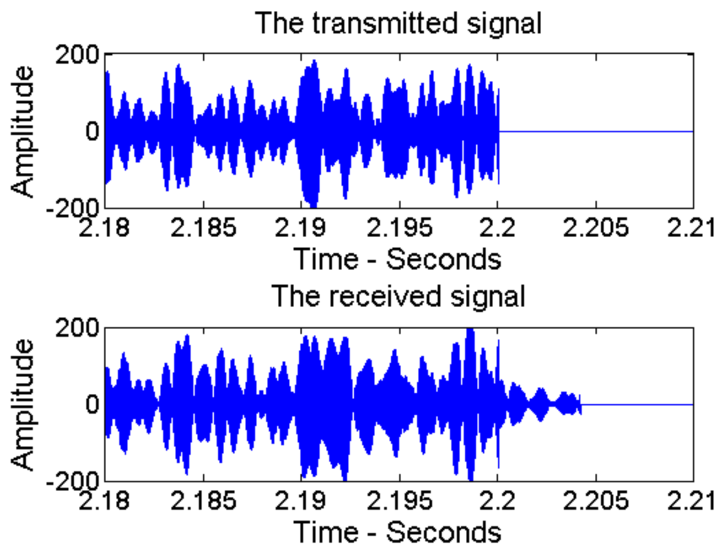


Figure 4.24: The transmitted signal and the received signal.

the echoes at the receiver. In figure 4.25 the results of the phase difference calculation of the received signal is plotted. This may look chaotic, but when

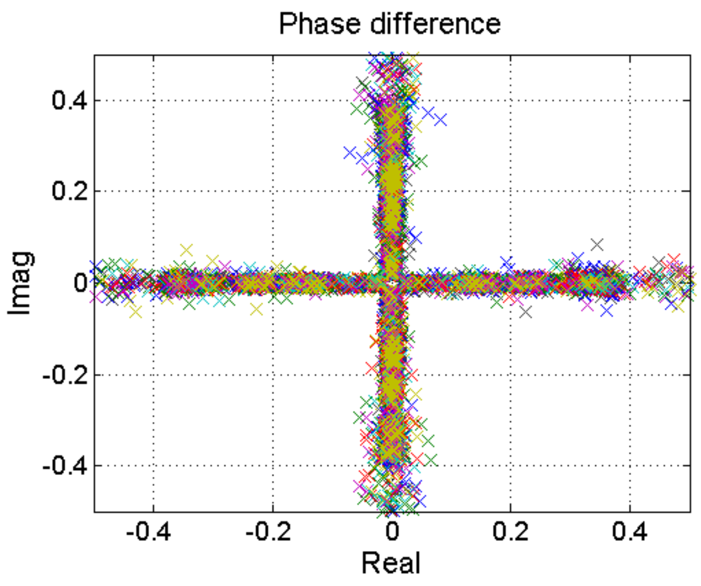


Figure 4.25: The phase difference with multipaths.

one takes a close look we see that though the amplitude varies quite much, the phase does not. This makes it possible for the decision operation to still make the right decisions. The resulting wave-file is shown in figure 4.26, and as expected it is without errors.

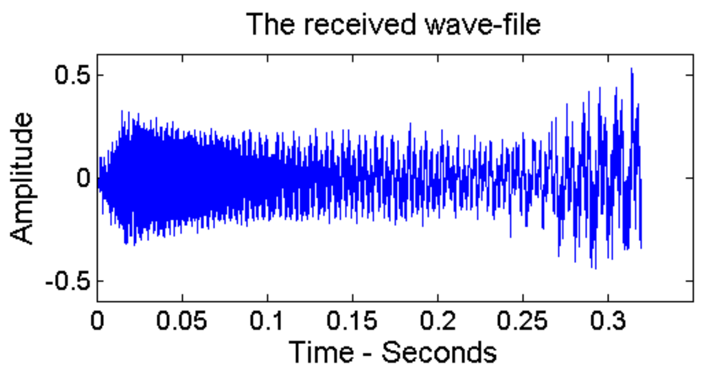


Figure 4.26: The received wave-file with multipaths.

Figure 4.27 shows the BER curves with and without multipaths. Here we see that the two curves follow each other closely until the SNR reaches about 16 dB. After 16 dB the bit error ratio with multipaths does not drop as quickly as the bit error ratio without multipaths. At a BER of order 10^{-4} the difference in SNR is about 1 dB for the two curves.

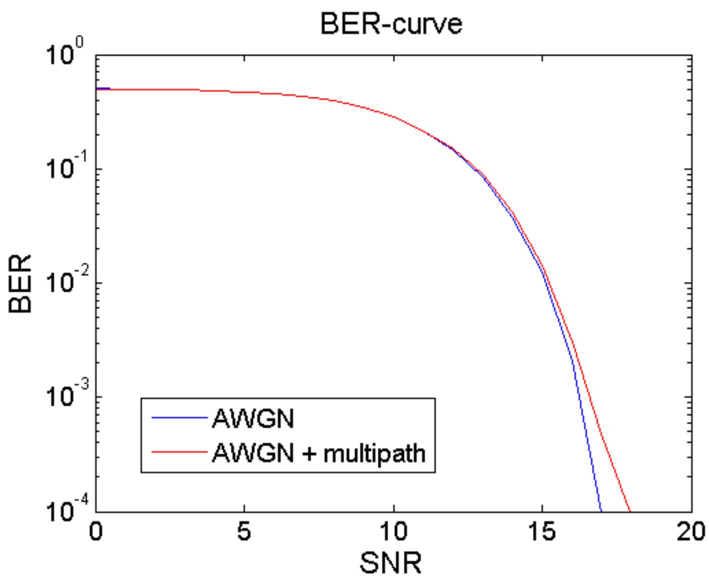


Figure 4.27: The bit error ratio as a function of SNR.

4.2.4 Synchronization error

When a signal travels through a complicated communication channel it can change quite drastically. This change can lead to an error in the synchronization at the receiver. The synchronization is important because of the FFT operation preformed at the receiver and it is done by correlating the reference symbol, which is known at the receiver, with the reference symbol received at the receiver. The start of the FFT-window is then found by locating the correlation top. Because of the complicated channel this correlation top is not always so obvious and the system may detect the wrong top. This can cause errors at the receiver after the FFT operation.

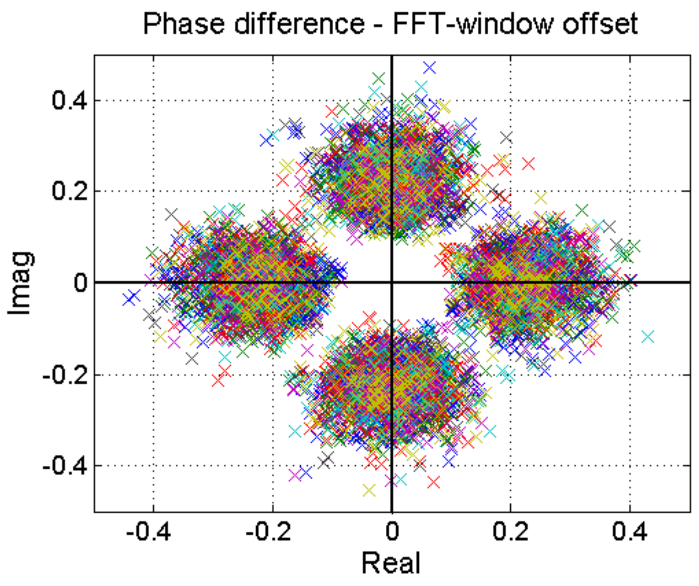


Figure 4.28: The phase differences with FFT-window offset.

In figure 4.28 the calculated phase difference before decision is shown. The FFT-window offset in this case is approximately 10 ms. Here we see that the effect of the FFT-window offset is quite similar to the effect of AWGN (see section 4.2.1). The variance of the phase differences has increased, but the mean are still located at i, -i, 1 and -1.

The bit error ratio is plotted against the FFT-window offset in time in figure 4.29. Here we see that the OFDM system is quite sensitive to this offset. A offset of about +/- 0.01 seconds gives no errors in the reception, but the bit error ratio increases rapidly for larger offsets. When the offset reaches +/- 0.2 seconds, depending on the guard interval, the FFT-window overlaps two symbols and as expected the bit error ratio is close to 0.5.

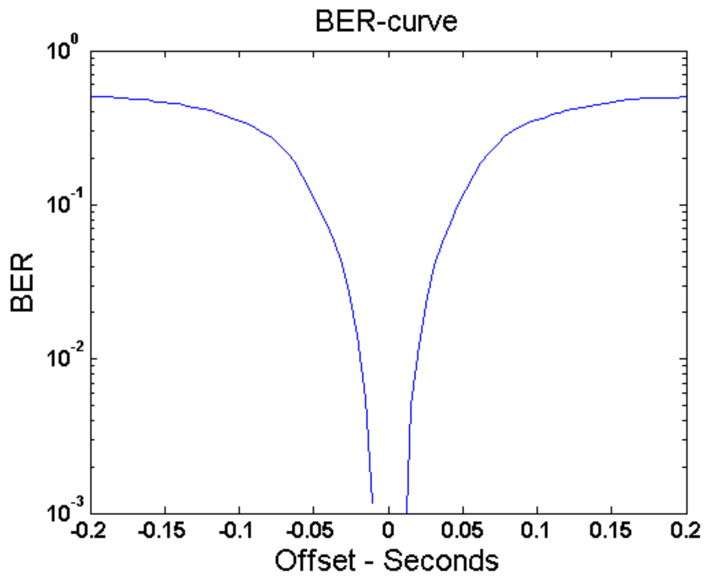


Figure 4.29: The bit error ratio as a function of offset in time.

4.2.5 A ”real” communication channel

In this section we will try to simulate a OFDM transmission with a ”real” communication channel. By ”real” we mean a channel composed of the elements described in sections 4.2.1, 4.2.2 and 4.2.3. The signal to noise ratio for the channel is 18 dB, the phase error for the channel is given by figure 4.19 and the multipaths are given in figure 4.23. By combining all of these we get a more realistic and challenging channel.

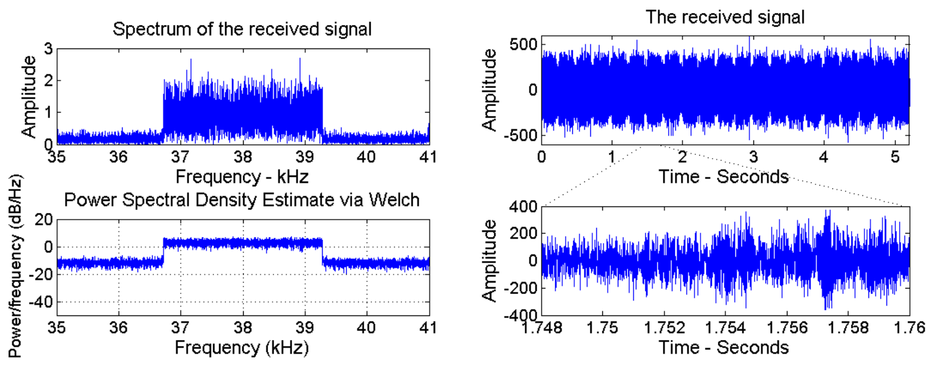


Figure 4.30: The received signal after channel influence.

In figure 4.30 the resulting received signal after the interaction with the channel is plotted in both the time and the frequency domain. We see from the frequency spectrum that the signal to noise ratio is 18 dB. The time domain signal to the right is now greatly distorted from its original shape. The noise together with the multipaths has now made it almost impossible to separate the guard interval from the OFDM-symbols and the periodic shape seen when enlarging the signal is no longer visible at the lower right in the figure.

The phase differences calculated at the receiver is plotted in figure 4.31. It can easily be seen that the resulting plot is a combination of the phase differences calculated for noise, phase error and multipaths, figures 4.16, 4.20, 4.25, respectively. From figure 4.31 we see that the decision operation now is quite difficult, and the probability for the wrong decision to be made has increased compared to the case in sections 4.2.1, 4.2.2 and 4.2.3.

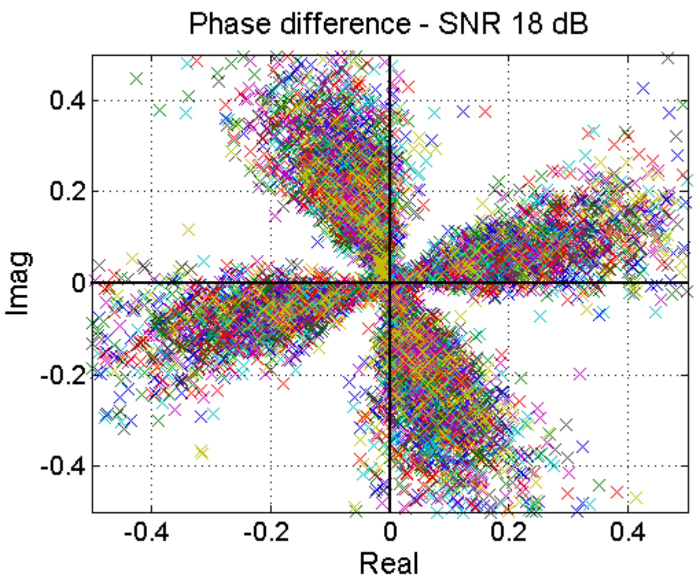


Figure 4.31: The calculated phase difference after channel influence.

The result after the decision operation is 86 bit errors out of a total of 20480 bits, resulting in a BER of 0.004. This means that in the worst case 86 out of a total of 2560 samples in the wave-file will turn out wrong. The resulting wave-file is plotted in figure 4.32 and some of the errors are clearly visible. The errors where the amplitude of the sample is increased results in a peak, while the errors where the amplitude decreases is not visible in the plot. The errors can be reduced by error coding or further processing at the receiver.

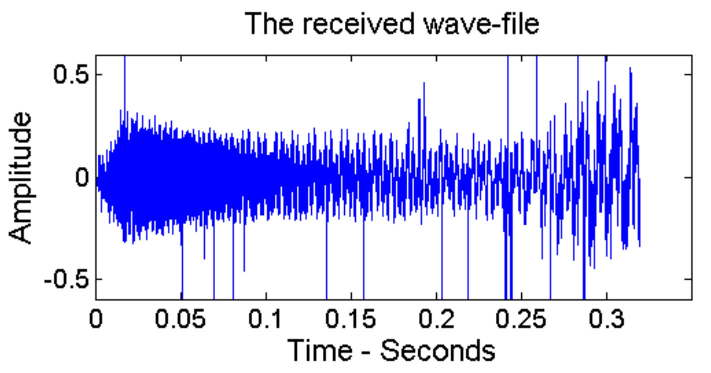


Figure 4.32: The retrieved wave-file at the receiver.

In figure 4.33 four bit error ratio curves are plotted. The blue curve is the BER when only noise is present on the channel, the red is the BER when noise and phase error is present and the green is the BER when noise and mulitpaths is present. The yellow curve is the bit error ratio for the channel when all of these are present. Here we see that the BER curve for this channel is approximately the sum of the blue, red and green curves. In order to achieve a bit error ratio in the order of 10^{-3} the signal to noise ratio must be 20 dB or more.

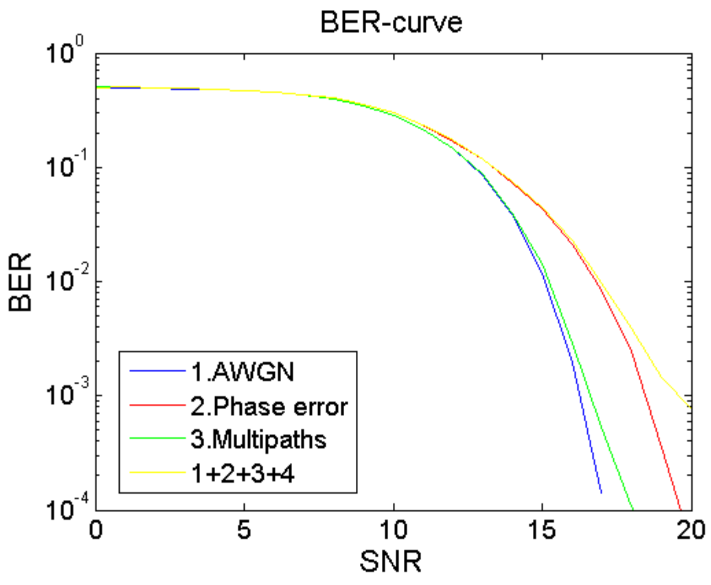


Figure 4.33: Bit error ratio for the communication channel.

Chapter 5

Initial measurements

In this chapter we describe an experiment that took place at Trondheim harbor on the 9th of May 2007. The preparations and the measurements itself was preformed by the author and Tor Arne Reinen from SINTEF. The objective was to test the OFDM-system described in chapter 3.

5.1 The area of interest

Before we describe the experiment, we will start by describing the area where the measurements took place. We will also take a closer look at the properties of the seabottom and the environment. The figures in this section is produced by NGU (Norwegian geological survey).

5.1.1 Locations

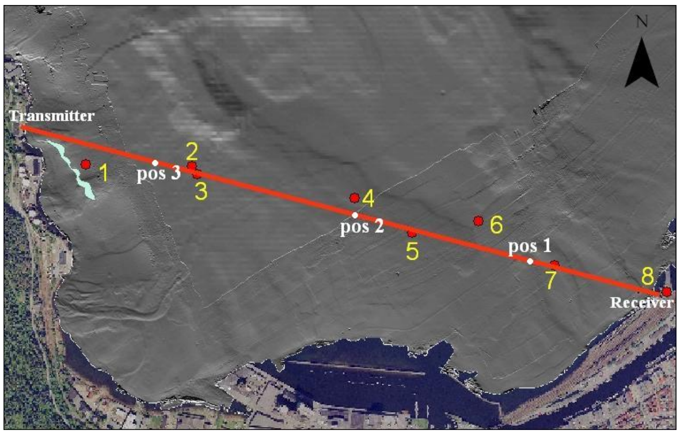


Figure 5.1: Trondheim Harbor

The locations of the transmitter and the receiver is as shown in figure 5.1. The transmitter is located at ”Trondhjem Biological Station” and the receiver is located at at a place called ”Sjøbadet”. The distance between the the transmitter and receiver is 2.1 km. The shipping activity in the area is moderate and consists of most small (e.g. speed boats) and medium (e.g. ferries) vessels.

5.1.2 The seabottom

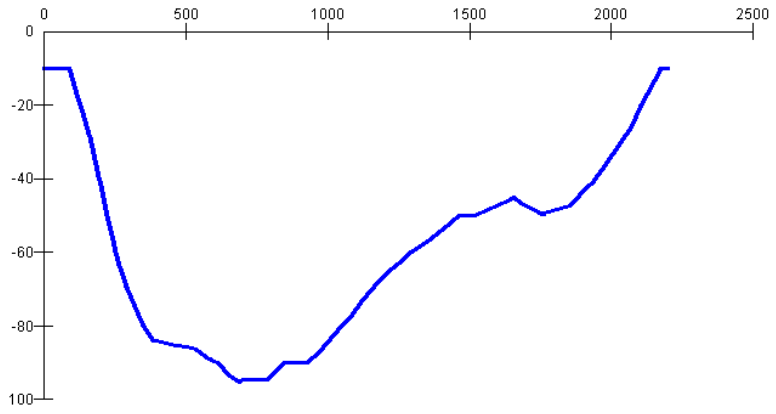


Figure 5.2: Profile of the seabottom

Figure 5.2 shows a profile of the seabottom from ”Trondhjem Biological Station” to ”Sjøbadet”. From the figure we can see that the depth varies quite a lot, from 4 to 95 meters. The 3D map of the bottom shown in figure

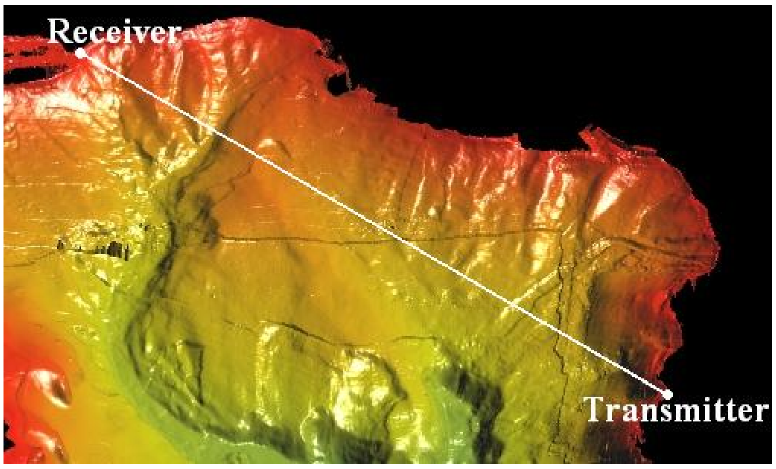


Figure 5.3: A 3D map of the bottom

5.3 shows that the depth is relatively constant perpendicular to the profile (the white line). This means that we can assume that there are no large structures located outside the path in figure 5.1 reflecting the signal back to the receiver. Because of this our channel is most likely not so much influenced by the sound propagating in other directions than toward the receiver. We can therefore assume that the bottom structures affecting our channel are those shown in figure 5.2.

In figure 5.1 there are eight red dots. Each of these dots represents a bore hole sample. These bore holes tells us something about the properties of the bottom. The data from all eight bore holes is summarized in table A.1 in appendix A. What we see inn this table is that the seabottom is made up by loose and unconsolidated sediment such as sand, silt and clay.

5.1.3 Environment

On the day of the experiment the wind was about 3 meters per second and there were moderate precipitation. The temperature was approximately 5 degrees celsius [2].

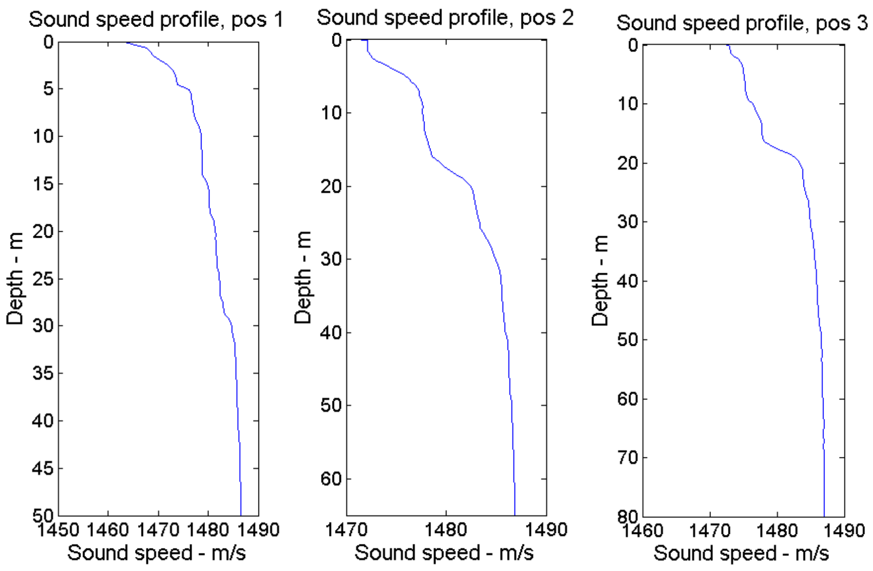


Figure 5.4: Sound speed profile measured 17.11.06

In the figure above three sound speed profiles are shown. These were measured 17th November 2006 by the NTNU research vessel, "Gunnerus". The location of the sound speed profiles are shown in white in figure 5.1. These were the only profiles available to us at this location and there is a good reason to believe that these profiles will differ some from the actual profile on the 9th of May 2007. But since the temperature in the water do not differ much from November to May we will assume that the profiles at the time of the experiment are similar to those in the figure. We see from figure 5.4 that the sound speed varies some at the upper layer at the three positions, but all three profiles shows that the sound speed in the water increases as a function of depth.

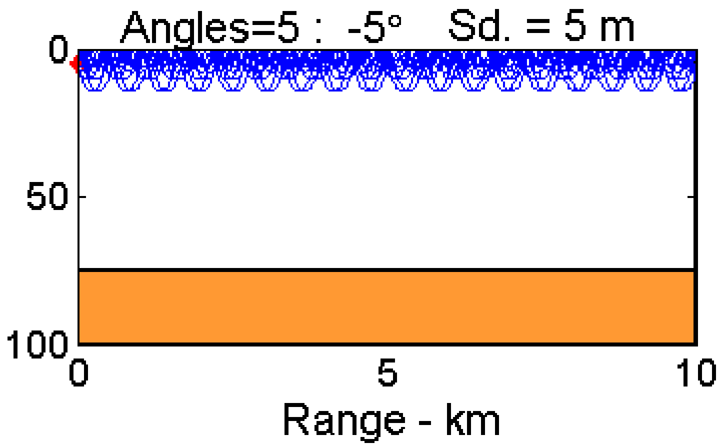


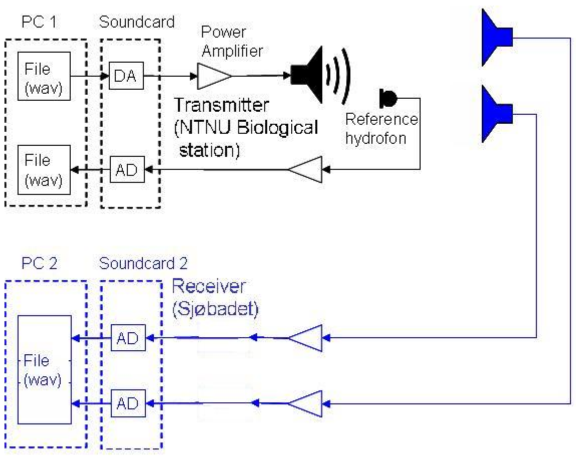
Figure 5.5: A ray trace with the sound speed profile a position 2.

A ray trace with the profile at position 2 is shown in figure 5.5. The plot is created by PlaneRay, which is a ray trace program developed by Jens M. Hovem at The Norwegian University of Science and Technology (NTNU). For details about the program PlaneRay view [5]. Here the source are placed at a depth of 5 meters with a opening angle of 10 degrees. From the figure we see that when the sound source are placed near the sea surface all the rays will bend upwards and be reflected at the surface. This will create a surface channel were all the sound propagate. This means that there will be many arrivals at the receiver which will cause a long channel impulse response.

5.2 Hardware setup

5.2.1 Transmitter

The setup for the transmitter is as shown to the upper left in figure 5.6. The transducer of type SKIPPER is mounted on a aluminum rack. It is pointing directly at the receiver 2 m below the surface. The depth of the water is 4 m. The directivity pattern for the transducer is shown in figure 5.7. The transducer is mounted so that it is most directive in the vertical



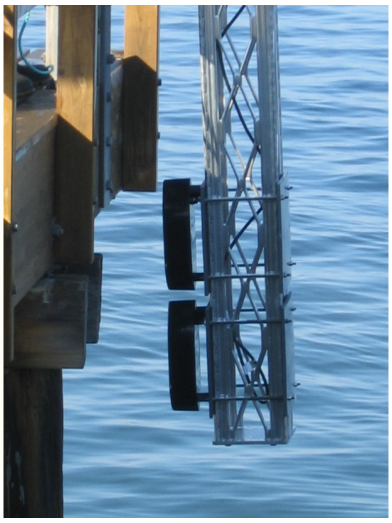


Figure 5.6: Left: The setup for the experiment. Right: Two transducers mounted on the aluminum rack at the receiver.

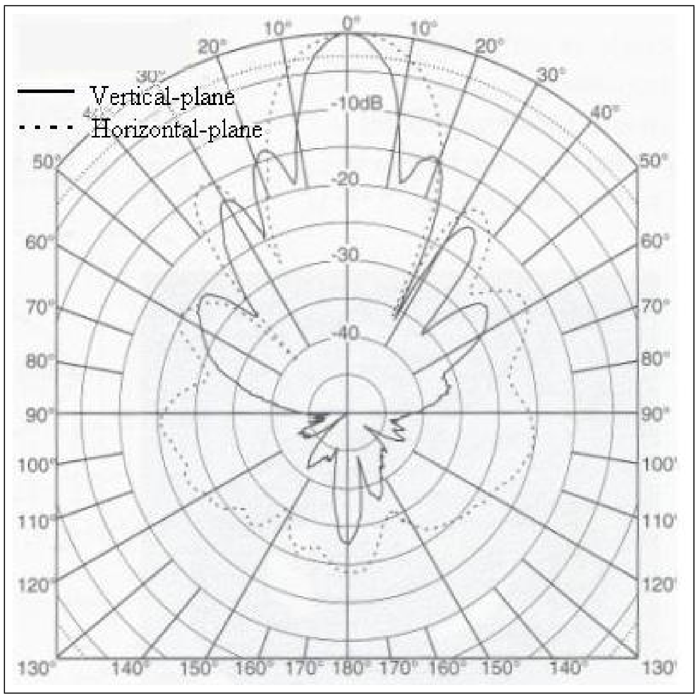


Figure 5.7: Directivity pattern for the transducer

plane, and it has a opening angle of approximately 10 degrees. The center frequency is at 38 kHz and the bandwidth is approximately 5 kHz. The transducer is connected to a power amplifier before it is connected to a sound card. The sound card, which preforms the D/A conversion, is attached to a laptop. The sound files we want to transmit are stored on this laptop. At the transmitter we have also mounted a calibrated reference hydrophone so that we can calculate how much power which is transmitted in to the water.

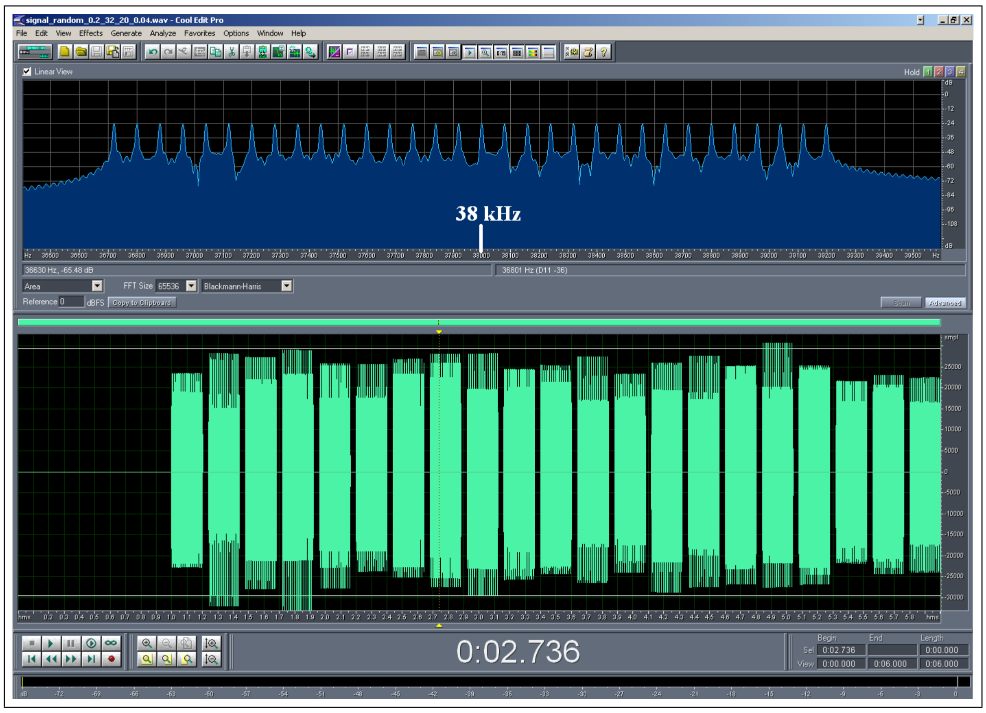
5.2.2 Receiver

The receiver side is very similar to the transmit side, and is shown in the lower left in figure 5.6. One difference is that we here have mounted two transducers of the same type as the one at the transmitter on a aluminum rack (Figure 5.6). The lower and upper transducer is located 2 m and 1.6 m respectively below the surface. The reason for using two transducer is that we then have the possibility to consider the space variation between the to received signals. Each of the two transducers are connected to a power amplifier before the signal travels through a sound card and is A/D converted. The received signal is then recorded on a laptop and stored for further processing.

5.2.3 Signal format

A total number of twelve signals are to be used in the experiment. These are OFDM signals with different parameters created by the OFDM program described in chapter 3. A overview and details of these signals are given in table B.1 in appendix B. The main parameters which are varied are the symbol length T_u, and the number of sub-carriers.

One of the experiment signals to be transmitted are shown in figure 5.8. The figure shows a screen shot of the program Cool Edit Pro, which will be used to playback the wave-files at the transmitter and to record the received signal at the receiver. The signal which is loaded in Cool Edit Pro is signal number one in table B.1. Here we see that the total number of OFDM symbols are 21, including the reference symbol. From the frequency analysis at the upper part of the figure we clearly see that the signal has 32 carriers centered around 38 kHz. The length of the signal is 5 seconds. In order to be able to play the signal in a loop during the experiment a one second silence has been added at the beginning of the signal. The reason for wanting to play the signal in a loop is to be able to detect if the channel changes drastically over time. The data rate for this signal is 256 bits per seconds, and a total number of 1280 bits will be transmitted.


signal_random_0.2_32_20_0.04.wav - Cool Edit Pro
File Edit View Effects Generate Analyze Favorites Options Window Help
Linear View
38 kHz
Hz
hms
0:02.736

36630 Hz, -65.48 dB
36801 Hz (D11 -36)
Area
FFT Size 65536
Blackmann-Harris
Reference 0 dBFS

	Begin	End	Length
Sel	0:02.736		0:00.000
View	0:00.000	0:06.000	0:06.000

Figure 5.8: Screen shot of one of the experiment signals (12 dB/div in the frequency plot)

5.3 Results

In this section we will present the results of the experiment described above. We will take a closer look at one of the received signals and show some figures.

In figure 5.9 the channel impulse response is plotted. This was measured 03.05.07, and has been calculated by SINTEF. Here we see that there are a large number of arrivals, and the length of the channel impulse response was approximately 20 ms. This long impulse response is as predicted by the ray trace in figure 5.5, which means that the sound speed profiles shown in figure 5.4 must have been similar to the actual profiles on the day of the experiment. Because of the long channel impulse response and the many strong arrivals this channel will be very challenging for a communication system.

The signals transmitted are those shown in table B.1 in appendix B. The recorded signal at the receiver had a length of approximately one minute. This means that the data was transmitted 10-12 times for each recording (duration of one transmission is approximately 5 seconds). All the received signals were stored on the computer at the receiver for further processing.

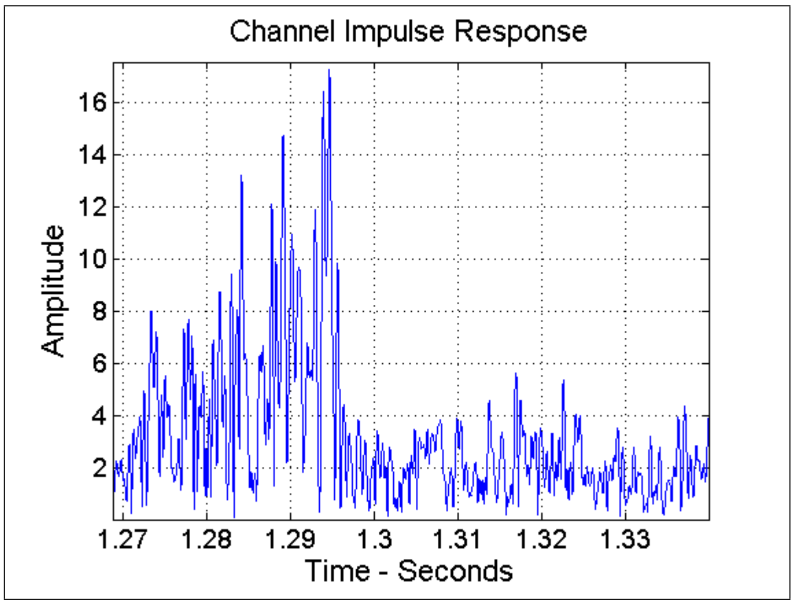


Figure 5.9: The channel impulse response.

After the measurements were completed the received signals were demodulated by the OFDM-system. This was done by copying one of the data transmissions from the long recording in to a wave-file and then use this file as input in the OFDM receiver. The result of the OFDM demodulating was disappointing. Almost all of the demodulations resulted in a bit error rate in the range of 0.4 to 0.5. This bit error rate is the same as if there had been a random detection at the receiver. In order to analyse what went wrong we are going to take a closer look at the demodulation of one specific signal. This is signal number 8 in table B.1 in appendix B. This signal has a symbol duration T_u, of 100 ms, and 128 sub-carriers. The guard interval is 20 ms. The total number of OFDM-symbols in the transmission is 41, including the reference symbol.

Figure 5.10 shows the signal as it appeared on the laptop at the transmitter. In the figure one can clearly see the 41 OFDM symbols separated by the guard interval. In the upper part of the figure one can see the frequency spectrum of the signal. The frequency spectrum is made up by 128 subcarriers centered around 38 kHz.

The measured signal at the receiver is shown in figure 5.11. In this recording there are 10 data transmissions. The two signals shown in the lower part of the figure are the recordings done by the upper and lower transducer, respectively. The effect of the channel is clearly visible at the frequency plot in the figure. By comparing the frequency plot in figure 5.11 with the

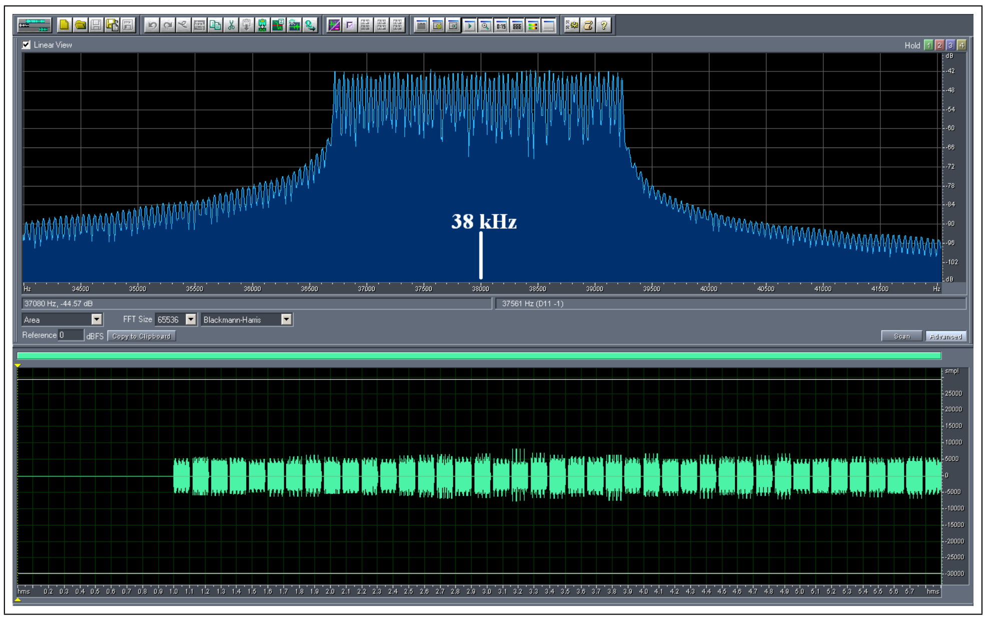


Figure 5.10: The transmitted signal (6 dB/div in the frequency plot).

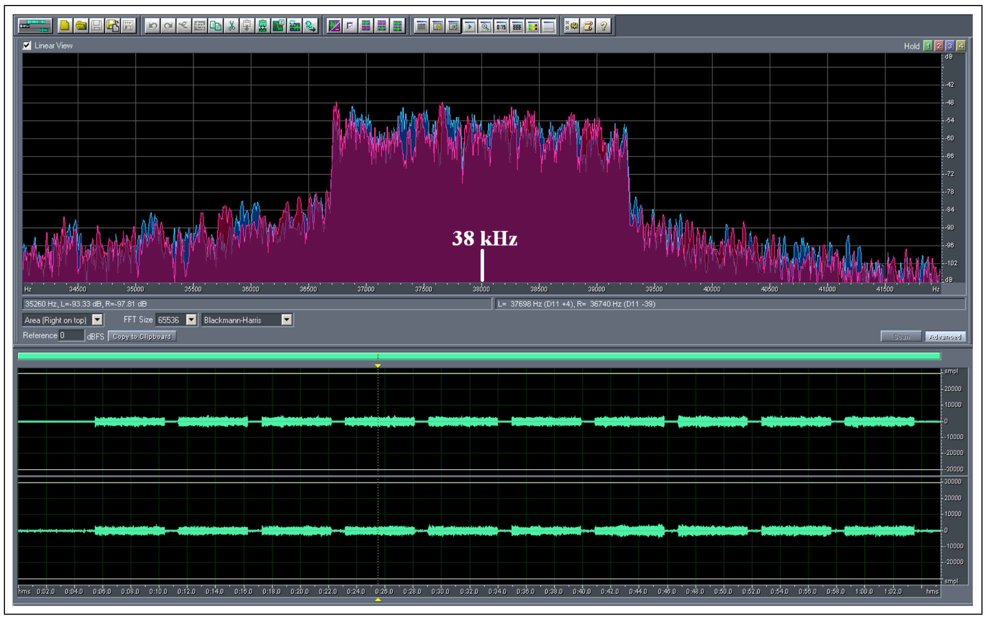


Figure 5.11: Received signal at ”Sjøbadet” (6 dB/div in the frequency plot).

frequency plot in figure 5.10 we see that the received signal, as expected, is more noisy than the transmitted. From the frequency plot we see that the signal to noise ratio (SNR) of the received signal is well within range of the demand specified in section 4.2.1 in chapter 4, so the noise should not cause any problems in the demodulation.

The short version wave-file used in the demodulation is shown in figure 5.12. In this figure the effect of the channel is also visible in the time domain. When we compared to the time domain signal in figure 5.10 we notice that the guard interval is no longer so distinct. This is due to multipaths in the channel, which creates a ”tail” in the first part of the guard interval. This is shown in the lower part of figure 5.12. The length of the ”tail” is approximately 20 ms, which is consistent with the length of the channel impulse response.

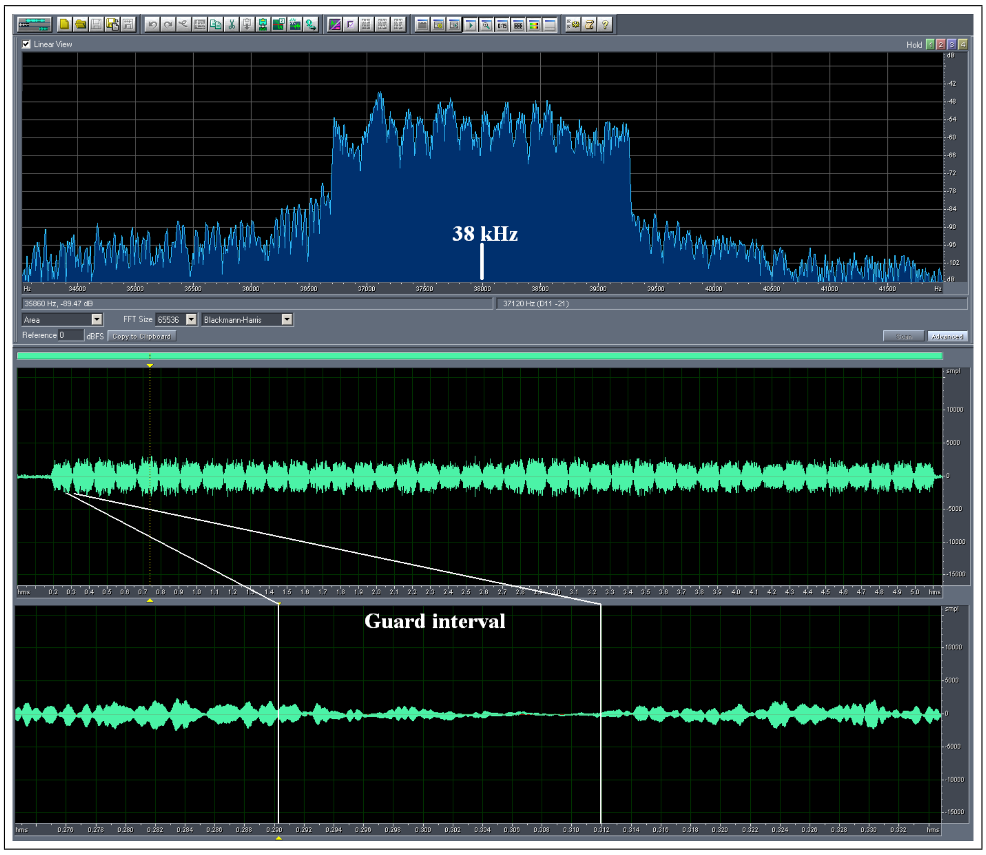


Figure 5.12: One transmission received at ”Sjøbadet” (6 dB/div in the frequency plot).

Figure 5.13 shows the correlation between the known reference symbol and the received signal. This is used to synchronize the demodulation. Here we see that the correlation function has a strong peak around sample number

38000. The demodulation will start at the sample which has the highest value in the correlation function. We also notice that the highest peak is not the first peak. This indicates that the strongest arrival is not the first arrival, which also is consistent with the channel impulse response in figure 5.9.

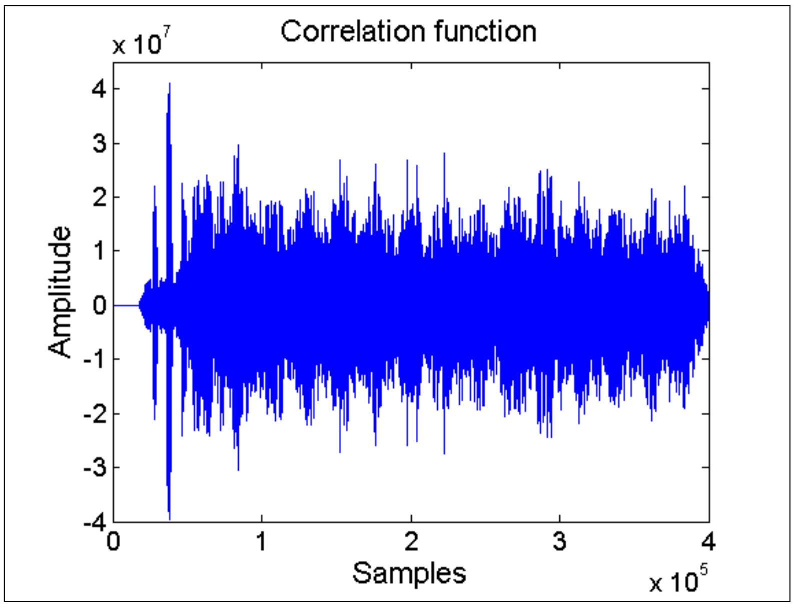


Figure 5.13: The correlation function of the reference symbol and the received signal.

The complex phase differences calculated in the differential decoder in the receiver is plotted in figure 5.14. We see that the phase difference now is more random than in the simulations in chapter 4. The complex values does not tend to gather around the values i, -i, 1 and -1 as we expected. This will lead to errors when we retrieve the binary data. The result after the demodulation is that out of 5120 QPSK symbols, 2728 turned out wrong. The number of bit errors was 3287 out of 10240. This results in a bit error ratio (BER) of 0.321. This is lower than 0.4 but just incidental because the QPSK-symbol error ratio is close to 0.5.

Because the calculated complex phase difference appeared so random in figure 5.14, we suspected that it was the differential decoding that failed. In order to decide if the phase difference is i, -i, 1 or -1 the phase error from one OFDM-symbol to another can not be larger than $\pi/4$. If the phase error is larger the wrong decision will be made and the differential decoding will fail. This phase error can be caused by the communication channel and/or

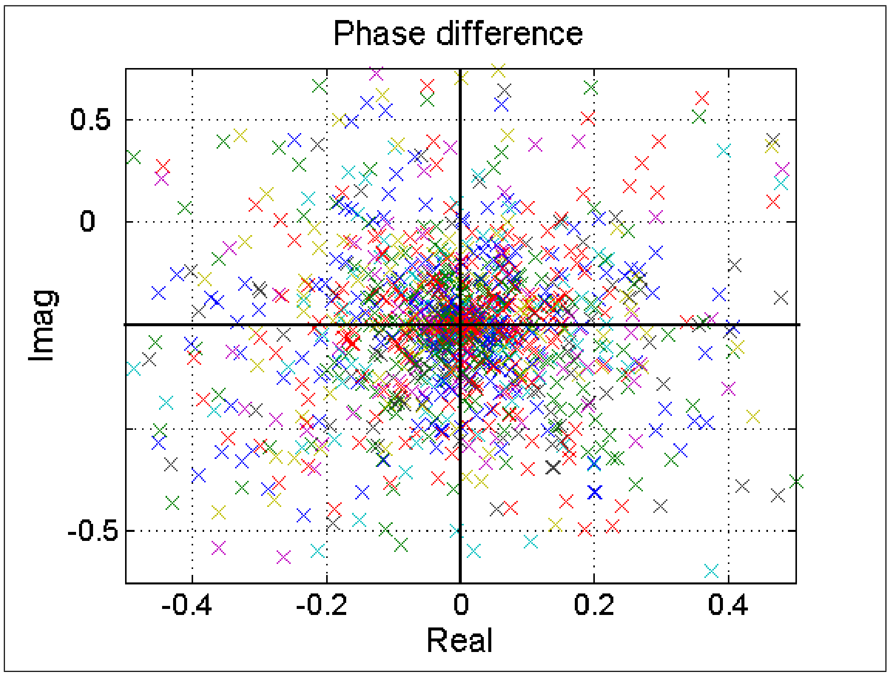
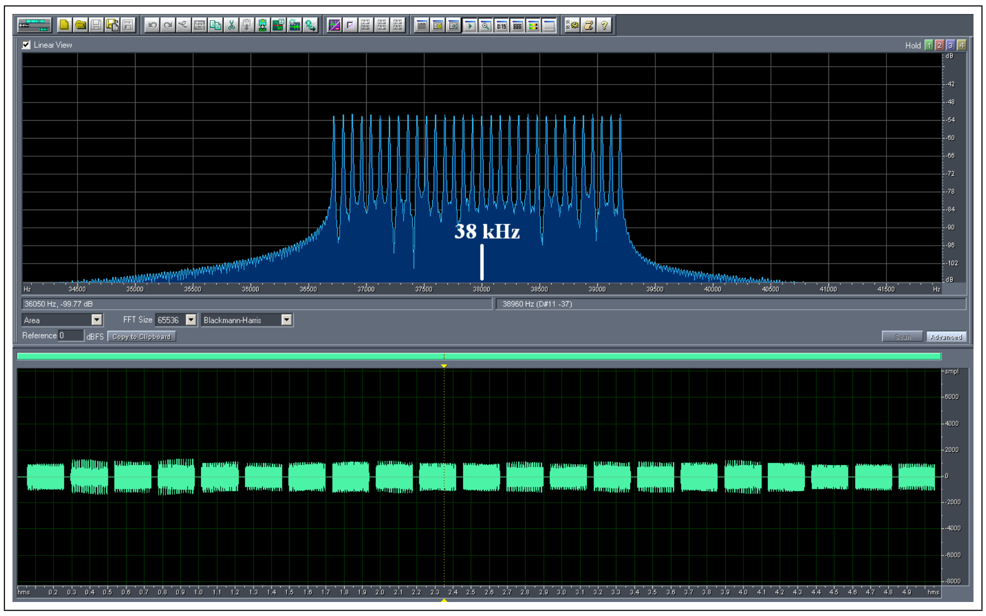


Figure 5.14: The calculated complex phase difference of the received signal.

oscillator drift in the equipment.

In order to check the oscillator drift in the equipment a simple experiment was preformed. The setup of the experiment was the same as shown to the left in figure 5.6, but instead of using the transducers, the soundcards at the transmitter and receiver was connected directly together. A signal with a symbol duration of 200 ms, 32 sub-carriers and a guard interval of 40 ms (see figure 5.8) was transmitted. The received signal is show in figure 5.15. Here we see that the noise in the signal is minimal, and the 32 sub-carriers are clearly visible in the frequency plot at the top. But even though the received signal is hard to separate from the original signal the result of the demodulation was a bit error rate (BER) of 0.5. This means that even under ideal conditions the system failed.

Figure 5.16 shows the complex phase difference calculated during the demodulation, together with the complex phase difference of the original signal (red). Here we see that the phase difference has rotated approximately 60 degrees to the right. This rotation is caused by a constant oscillator drift in the equipment and causes the demodulation of the received signal to fail. This means that even if the acoustic channel were noise free and there were no multipaths, the demodulation would still fail.



36050 Hz, -99.77 dB

38960 Hz (D#11 -37)

Area

FFT Size 65536

Blackmann-Harris

Reference 0 dBFS

Figure 5.15: The received signal in the simple experiment (6 dB/div in the frequency plot).

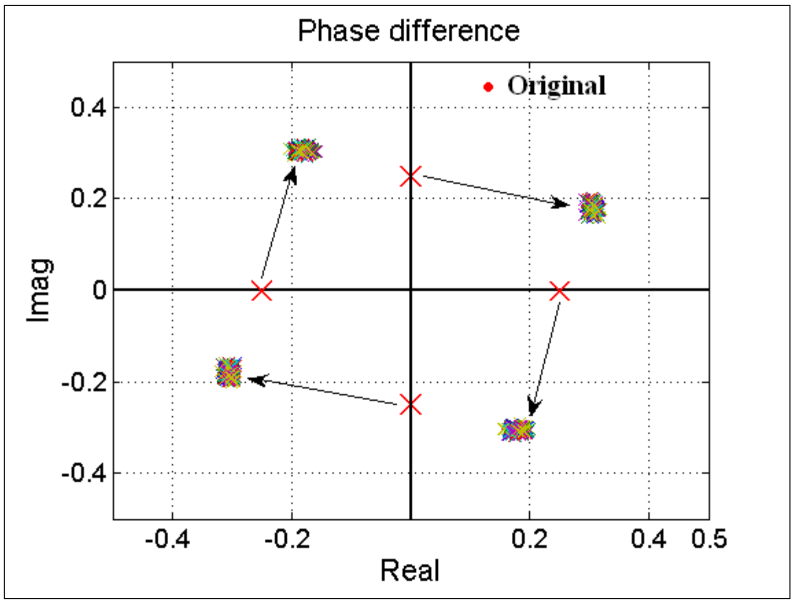


Figure 5.16: The calculated complex phase difference of the received direct signal.

By running simulations of the OFDM-system we were able to recreate the effect of the oscillator drift. This were done by adding a phase error to the signal until the resulting complex phase difference were identical to the one shown in figure 5.16. The phase error added in order to achieve that result

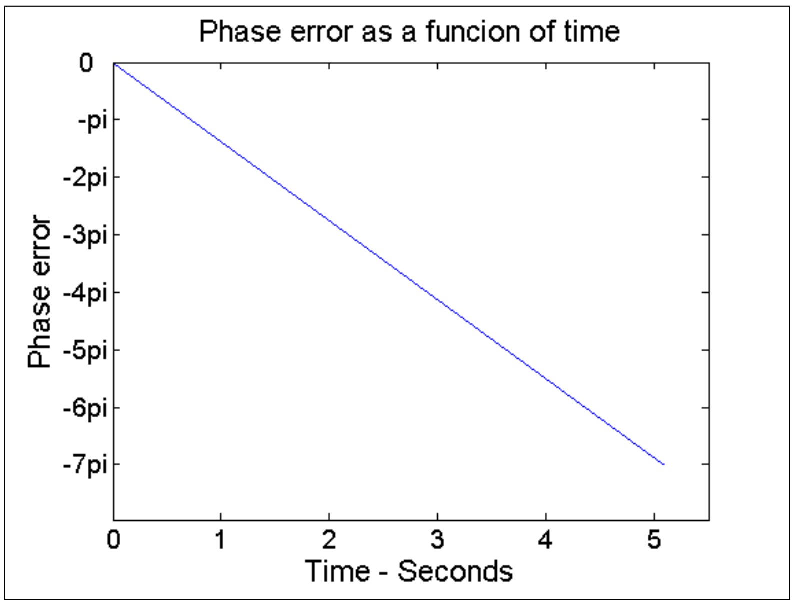


Figure 5.17: The simulated oscillator drift.

is shown in figure 5.17. As expected the constant oscillator drift can be simulated by a linear decreasing phase error. The phase error starts out at zero and ends up at -7π. In order to correct this oscillator drift at the receiver we simply add the same phase error to the received signal as the one shown in figure 5.17 but now with a positive amplitude. The resulting complex phase difference in the differential decoder when this is done is shown in figure 5.18. Here we see that the phase difference has been rotated back by approximately 60 degrees and the demodulation is now without any bit errors.

Since there were a oscillator drift present in the simple experiment, it must also been present during the measurements at Trondheim harbor. In order to check if this could improve the results we tried adding the same phase error as the one described above to the received signal in figure 5.12. The result of this was a improved bit error ratio (BER) from 0.321 to 0.300, which shows that it had some effect but not nearly enough.

The reason the OFDM-system failed in these measurements could be many. However we can narrow it down a bit. As shown in figure 5.12 the signal to noise ratio should be more than sufficient for the system to work, so we can

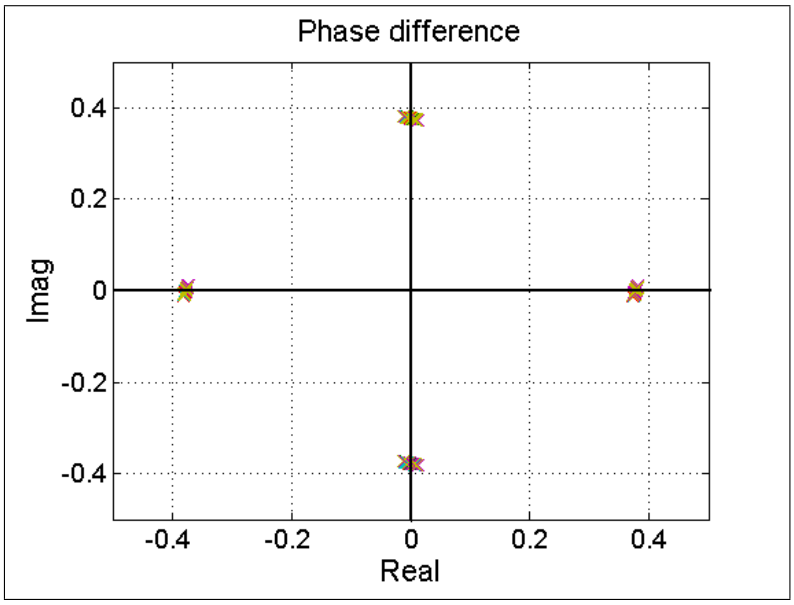


Figure 5.18: The calculated complex phase difference with oscillator drift correction.

probably assume that noise is not the problem. The length of the impulse response of the channel was a bit longer than expected, but by choosing the right OFDM parameters we should have been able to avoid this problem. The most probable reason the demodulation failed is phase error. Because the complex phase difference in figure 5.14 appeared so random it indicates that it is the differential decoding which failed. This means that the phase error from symbol to symbol must have been more than 45 degrees even when the oscillator drift has been corrected. This tells us that the phase error in the acoustic channel may have been to challenging for the differential coding to handle. It should also be mentioned that the oscillator drift in the simple experiment could differ some from the oscillator drift at the experiment at Trondheim harbor. This is because some of the conditions, e.g temperature and humidity, were not the same for the two experiments.

Chapter 6

Discussion and conclusions

6.1 The system

There are many methods of modulations used in wireless communications. In this study the aim was to develop a system for wireless underwater communications. The challenges of a underwater channel are many. The slow, depth varying propagation speed of the acoustic waves could cause the impulse response of the channel to be long with possibly many strong arrivals. Orthogonal frequency division multiplexing (OFDM) was chosen as the modulation technique because of its way of handling multipaths. OFDM is a multi carrier modulation technique. This means that the channel bandwidth is divided into several sub-bands. By letting each sub-band have a relatively low data rate one is able to maintain a symbol duration much longer than the impulse response of the channel, while still achieving a high data rate totally. It is this long symbol duration together with a guard interval that reduces the intersymbol interference and therefor reduces the effects of multipaths considerably.

In order to reduce the complexity of the system but still maintain a high data rate, differential phase shift keying (DPSK) was chosen as the coding technique. By using differential coding we avoided a complex carrier tracking at the receiver because the data now lies in the phase difference of two succeeding symbols, not in the received symbol itself. The differential coding also prevents to some extent Doppler shifts from being a problem at the receiver. Quadrature phase shift keying was chosen in order to be able to send two bits per sub-carrier which increased the total data rate of the system.

The OFDM-system developed in this study has been thoroughly tested through simulations. It has been tested using several different communications channels containing noise, multipaths, phase error or a combination of these. The effects of synchronization error has also been demonstrated. The results of these simulations shows that the OFDM-system handles these challenges quite well. When only noise is present on the channel the requirement of the signal to noise ratio (SNR) is 17 dB or higher (BER in the order 10^{-4} or less). When also adding a linear phase error with a maximum of 2π, we see that the requirement of the SNR increases by approximately 3 dB. But the most interesting part was to observe how the system handles multipaths, since this should be the strength of the OFDM-system. The simulated channel impulse response consisted of five arrivals, where four were relatively strong reflections. The length of the impulse response was approximately 4 ms. In the results we see that when noise and multipaths were present on the channel the requirement of the SNR was 18 dB, an increase of only 1 dB compared to a channel containing only noise. A possible weakness of the OFDM-system was also observed during the simulations. This was that the OFDM-system was relatively sensitive to synchronization errors at the receiver.

The use of a OFDM-system for underwater communications is very promising. Its ability to handle multipaths while still be able to maintain a high data rate is the main reason for this. By also using DQPSK the complexity is decreased while the data rate is increased even more. The simulations on the computer showed that the system could deal with different challenging channels parameters, with satisfying results. By reducing the number of sub-carriers used, the requirements of the system could be decreased even more. The differential coding is a very efficient way of reducing the complexity and to handle Doppler shifts. However this is also a quite vulnerable coding, especially because of the long symbol duration used in OFDM. If there is a large phase drift in either the equipment or the channel, or if the Doppler shift is to large, the differential coding will fail. When simulating on the computer this was not a problem, but it is very difficult to say how these conditions will behave on a real communication channel.

6.2 Initial measurements

Based on the OFDM-system developed in this study initial measurements have been preformed in a real underwater environment. The measurements where preformed in a area with many challenges. The bottom topography was very curved and the depths ranged from 4 meters to 95 meters. This is a challenging communication channel, but one that is quite common for harbor areas. The channel impulse response was very long with many strong arrivals, which would make communications more difficult. The various signals used in the experiment were created by the OFDM-system.

When the signals received were processed we discovered that the demodulation of the OFDM signals failed. After analysing the demodulation we found that the most probable reason for this was that the differential coding had failed. By preforming the same experiment, but without transmitting on the acoustic underwater channel, we discovered that there were a oscillator drift in the equipment used. The phase error created because of the drift was so large that the demodulation failed even without the acoustic channel. By running simulations on the computer the phase error caused by the drift was classified and corrected in the simple experiment. An attempt to correct this on the measurements preformed on the underwater acoustic channel was also tried, but without any significant improvement. The reason for this could be that the oscillator drift were not the same under the two experiments, and/or that the phase error in the channel was to severe for the system to handle.

Though the result of the initial measurements were not what we had hoped for, valuable experience was gained. Important knowledge was gained about the equipment used and about some of the issues in the OFDM-system. Because of the oscillator drift in the equipment and the challenging acoustic channel, other methods should be considered instead of the differential coding. This could for instance be a completely none-coherent method such as frequency shift keying (FSK) or a coherent method involving carrier tracking.

Bibliography

[1] S. C. Clay and H. Medwin. *Acoustical oceanography: Principles and applications*. John Wiley & Sons, 1977.

[2] Trondheim Harbor. Weather. *http://www.trondheim.havn.no/default.aspx?tabid=151*, 17.05.2007.

[3] Simon Haykin. *Communication Systems*. John Wiley & Sons, Inc, 2001.

[4] Jens M. Hovem. *Marine Acoustics, The Physics of Sound in Underwater Environments*. 2005.

[5] Jens M. Hovem. PlaneRay: An acoustic underwater propagation model based on ray tracing and plane wave reflection coefficients. 2006.

[6] Louis Litwig and Michael Pugel. The principles of OFDM. 2001.

[7] Richard Van Nee and Ramjee Prasad. *OFDM for Wireless Multimedia Communications*. Artech House, 2000.

[8] Ashish Pandharipande. Principles of OFDM. *IEEE*, 2002.

[9] Fredrik Tufvesson. Channel related optimization of wireless communication systems. Master's thesis, Lund University, Department of Applied Electronics, 1998.

Appendix A

Bore hole data from NGU

Table A.1: Data from the boreholes

Borehole	Start d	L1 Soil	d L1	L2 Soil	d L2	L3 Soil	d L3	L4 Soil	d L4	L5 Soil	d L5	L6 soil	d L6	L7 Soil	d L7
1	55	Sandy silt	4.0	Sandy clay	4.5	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
2	88	Fine sand	0.1	Silty clay	1.5	Clayey silt	0.5	Sandy silt	2.0	NA	NA	NA	NA	NA	NA
3	82	Sandy silt	0.4	Silt	2.0	Sandy silt	0.7	NA	NA	NA	NA	NA	NA	NA	NA
4	68	Fine sand	1.0	Sandy silt	2.8	Sand	0.5	Silty sand	0.8	Sand	0.8	Silty sand	2.1	NA	NA
5	50	Sand	0.3	Silty sand	0.8	Sandy silt	3.0	Clayey silt	1.9	Silty sand	1.0	Silty clay	1.0	Sandy silt	1.0
6	50	Sandy silt	1.2	Clayey silt	0.8	Sandy silt	0.8	Clayey silt	2.2	Silty clay	3.3	NA	NA	NA	NA
7	45	Silty sand	2.0	Silt	0.4	Fine sand	5.6	NA	NA	NA	NA	NA	NA	NA	NA
8	5.8	Gravelly sand	5.5	Coarse sand	2.5	Silty sand	2.5	Gravelly sand	2.0	Sand	2.5	NA	NA	NA	NA

In table A.1 the data from 8 different bore holes samples located between "Trondhjem Biological Station" and "Sjøbadet" is presented. The exact position can be seen in figure 5.1. The year the samples were collected ranges from 1953 to 2004. The depth of the bore holes varies from 4.1 meters to 15 meters. From the table we can see that the bottom is made up of mostly loose and unconsolidated materials such as sand, silt and clay.

Appendix B

Measurments

Table B.1: The experiment signals

Signal	B (Hz)	T_u (ms)	Carriers	OFDM symbols	Guard (ms)	Bits	Bits/s
1	2560	200	32	20	40	1280	256
2	2560	200	64	20	40	2560	512
3	2560	200	128	20	40	5120	1024
4	2560	200	256	20	40	10240	2048
5	2560	200	512	20	40	20480	4096
6	2560	100	32	40	20	2560	522
7	2560	100	64	40	20	5120	1045
8	2560	100	128	40	20	10240	2090
9	2560	100	256	40	20	20480	4180
10	2560	50	32	80	10	5120	1056
11	2560	50	64	80	10	10240	2111
12	2560	50	128	80	10	20480	4223

Table B.1 gives an overview of the signals used in the experiment. Twelve different signals with a bandwidth of 2560 Hz were made. Three different symbol length T_u, were used. The number of carriers used varies from 32 and up to the highest number were the carrier spacing still is orthogonal. This depends on the symbol length. The guard interval was set to one fifth of the symbol length, T_u [7]. As a result of the varying number of carriers and symbol lengths the data rate is also varying, as can be seen to the right in the table. The data type in all the signals are ”random”.

Appendix C

Matlab-code

OFDMtransmitter.m

```
%The function used to generate a OFDM signal.

function s_tot = OFDMtransmitter(symbollength, bandwidth, guard, data_type,
numbofcarriers, numberofsymbols, plots)

B = bandwidth;                    %Bandwidth
N = numbofcarriers;               %Number of carriers
Tu = symbollength;                %Symbol length
freq_space_factor = round(B*Tu/N); %Factor spacing of the frequency spectrum
FS=192e3;                         %Sampling freq
T = 1/FS;                         %Elementary period
fc = 38e3;                        %Carrier freq
guard = guard;                    %Guard time
guard_samples = round(guard/T);   %Guard time in samples
tt = 0:T:Tu;                      %Time vector
t_guard = 0:T:(Tu+guard)  ;       %Time vector including guard time
M = numberofsymbols;              %Number of OFDM symbols - reference symbol
f = (1:FS);                       %Frequency vector
ttotal= 0:T:(Tu*(M+1)+M*T+(M)*guard); %Total time including delay and guard

%TRANSMISSION

%Generating binary data
if strcmp(data_type, 'sound');
    %Generates data based on a wave file
    [Data1 DataIn Fs_sound bits] = loadSound('Sound.wav', M*N*2, plots);
elseif strcmp(data_type, 'random');
    %Generates random data
    rand('state', 0);
    DataIn = round(rand(2*N*M,1));
end

%Generating QPSK data
Data_QPSK = QPSKmodulator(DataIn, N, M);

%Seral to parallell data
Data_par = zeros(N,M);
k = 1;
m = 1;
```

```
for nn = 1:length(Data_QPSK);
    Data_par(k,m) = Data_QPSK(nn);
    if k == N;
        k = 1;
        m = m + 1;
    else
        k = k + 1;
    end
end

%DPSK
%Creating a random reference symbol
rand('state',1)
bb = -1+2*round(rand(1,N)).'+i*(-1+2*round(rand(1,N))).';

%Mapping the symbols as a phasedifference
phase_shift = zeros(N, M);
Data_phase_diff = zeros(N, M+1);
Data_phase_diff(1:N,1) = bb./sqrt(2);       %Normalize the reference symbol
dim = size(Data_phase_diff);

for pp = 2:dim(2);
    for qq = 1:dim(1)
       if Data_par(qq,pp-1) == 1+i;
           phase_shift(qq,pp-1) = -1;
       elseif Data_par(qq,pp-1) == -1-i;
           phase_shift(qq,pp-1) = 1;
       elseif Data_par(qq,pp-1) == -1+i;
           phase_shift(qq,pp-1) = i;
       elseif Data_par(qq,pp-1) == 1-i;
           phase_shift(qq,pp-1) = -i;
       end
    %creating a vector where the phasedifference is b-vector
    Data_phase_diff(qq,pp) =(1/abs(phase_shift(qq,pp-1)))*phase_shift(qq,pp-1) *
    (1/abs(Data_phase_diff(qq,pp-1)))*Data_phase_diff(qq,pp-1);
    end
end

%Zero padding
A = size(Data_phase_diff);
info = zeros(length(tt), M+1);

for zz = 1:M+1
info(1:freq_space_factor:(freq_space_factor*(N/2)),zz) = [Data_phase_diff(1:N/2,zz)];
info(length(tt)-(freq_space_factor*(N/2)-1):freq_space_factor:length(tt),zz) =
[Data_phase_diff((N/2+1):A(1),zz)];  %Zeros placed in the middle of the data
end

%Creating M+1 OFDM symbols
for symbol = 1:M+1;

%OFDM
carriers = FS.*ifft(info(1:length(info),symbol));

%Upshifting
s_tilde = carriers.*exp(i*2*pi*fc*tt');

%Inserting guard
s_guard = zeros(1, length(t_guard));
s_guard((guard_samples+1):length(s_guard)) = s_tilde;
%Creating the total signal
s_tot((1+(symbol-1)*length(s_guard)):length(s_guard)*symbol) = s_guard;
```

```
%Saving the reference symbol
if symbol==1;
   ref = real(s_tilde);
end

end

 s = real(s_tot(guard_samples+1:length(s_tot)));

%Creating wave file
MAX_SN = max(abs(s_tot));
wavwrite(s/MAX_SN, FS, 'signal');

%Saving variables
if strcmp(data_type, 'sound');
    save('OFDMspace.mat', 'ref', 'Data_par', 'DataIn', 'N', 'M', 'B', 'FS', 'fc',
    'Fs_sound', 'bits', 's', 'T', 'Tu', 'f', 'guard', 'guard_samples', 'data_type',
    'plots', 'freq_space_factor');
elseif strcmp(data_type, 'random');
    save('OFDMspace.mat', 'ref', 'Data_par', 'DataIn', 'N', 'M', 'B', 'FS', 'fc',
    's', 'T', 'Tu', 'f', 'guard', 'guard_samples', 'data_type', 'plots',
    'freq_space_factor');
end

if plots == 1

%Plots

if strcmp(data_type, 'sound');
%Plot of soundIn and the binary stream
t = 0:1/Fs_sound:(length(Data1)-1)/Fs_sound;
figure();
subplot(211)
plot(t, (Data1-(2^(bits-1)))/(2^(bits-1)));
title('Wave-file to be transmitted')
axis([0 0.35 -0.6 0.6]);
xlabel('Time - Seconds')
ylabel('Amplitude')
subplot(212)
stem(DataIn(1:24));
title('Bit stream DataIn ');
axis([0 24 -0.5 1.5]);
set(gca,'ytick',[0 1]);
xlabel('Bit number');
ylabel('Bit value');
else
figure()
stem(DataIn(1:24));
title('Bit stream DataIn ');
axis([0 24 -0.5 1.5]);
set(gca,'ytick',[0 1]);
xlabel('Bit number');
ylabel('Bit value');
end

%Plot of the QPSK symbols
figure();
subplot(211);
stem(real(Data_QPSK(1:12)));
title('The real part of the Data\_QPSK-vector');
axis([0 12 -1.5 1.5]);
```

```
set(gca,'ytick',[-1 1]);
xlabel('Symbol number');
ylabel('Value');
subplot(212);
stem(imag(Data_QPSK(1:12)));
title('The imaginary part of the Data\_QPSK-vector');
axis([0 12 -1.5 1.5]);
set(gca,'ytick',[-1 1]);
xlabel('Symbol number');
ylabel('Value');

%Plot of the QPSK symbols
figure();
plot(Data_QPSK,'LineStyle','none', 'Marker','x', 'MarkerSize',10);
title('QPSK symbol constallation');
axis([-1.5 1.5 -1.5 1.5]);
grid on;
xlabel('Real');
ylabel('Imag');

%Plot of the received phase shifts before decision
figure();
plot(phase_shift(1:N,1),'LineStyle','none', 'Marker','x', 'MarkerSize',10);
title('Phase difference');
axis([-1.5 1.5 -1.5 1.5]);
grid on;
xlabel('Real');
ylabel('Imag');

%Plot of the baseband spectrum
figure();
subplot(211);
plot(f/1000, abs(fft(carriers, FS))/FS);
title('Spectrum of the carriers-vector');
axis([0 192 0 3]);
xlabel('Frequency - kHz');
ylabel('Amplitude');
subplot(212);
pwelch(carriers,[],[],[],FS);

%Plot of the frequency spectrum of s(t)
figure();
subplot(211);
plot(f/1000, abs(fft(real(s), FS))/FS);
title('Spectrum of s(t)');
axis([35 41 0 3]);
xlabel('Frequency - kHz');
ylabel('Amplitude');
subplot(212);
pwelch(real(s),[],[],[],FS);
axis([35 41 -50 20]);
end
```

loadSound.m

```
%The function used to convert a wave-file into binary data

function [Data1 bit_vector, Fs, bits] = loadSound(filename, numberOfsamples, plots)
[DataIn, Fs, bits] = wavread(filename);
```

```
DataIn3 = round((DataIn*128+128).');
Data1 = DataIn3(1:numberOfsamples/bits);

%Creating the sound to transmitt
wavwrite((Data1-(2^(bits-1)))/(2^(bits-1)), 'soundIn.wav')

%Decimal to binary
DataIn2 = dec2bin(Data1);

%Creating continuos bit vector
dim = size(DataIn2);
bit_vector = zeros(dim(1)*dim(2), 1);
teller = 1;
for j = 1:dim(1)
    for k = 1:dim(2)
        bit_vector(teller) = str2num(DataIn2(j, k));
        teller = teller + 1;
    end
end
```

QPSKmodulator.m

```
%The function used to map two by two bits on to a QPSK-symbol

function [b] = QPSKmodulator(bit_vector, N, M)

a_d = bit_vector;
b = zeros(N*M,1);
k = 1;

%Mapping two by two bits on to a QPSK-symbol
for j = 2:2:length(a_d);

    if a_d(j-1)==1 & a_d(j)==1;
        b(k) = 1 + i;
    elseif a_d(j-1)==0 & a_d(j)==1;
        b(k) = -1 + i;
    elseif a_d(j-1)==1 & a_d(j)==0;
        b(k) = 1 - i;
    elseif a_d(j-1)==0 & a_d(j)==0;
        b(k) = -1 - i;
    end
    k = k + 1;
end
```

OFDMreceiver.m

```
%The function used to demodulate the OFDM-signal created by the OFDMtransmitter.m

function biterror = OFDMreceiver()

clear all;
load OFDMspace.mat;
load MAX_SN.mat;

B = B;                          %Bandwidth
N = N;                          %Number of carriers
```

```
Tu = Tu;                    %Symbol length
FS=FS;                      %Sampling freq
T = 1/FS;                   %Elementary period
fc = fc;                    %Carrier freq
tt = 0:T:Tu;                %Time vector
M = M;                      %Number of symbols - reference symbol
f = (1:FS);                 %Frequency vector
ttotal= 0:T:Tu*(M+1)+M*T;   %Total signal time

%Reference vector
Data_par = Data_par;

%RECEPTION
max_sn = MAX_SN;
% ref_normalized = REF;

[m_normalized d] = wavread('signal');
m = max_sn.*m_normalized;
% ref = max_sn.*ref_normalized;

%Removing delay and synchronizing
m_sync = zeros(length(m_normalized),1);
m_short = m(1:400000);
x = xcorr(ref, m_short);
[C I] = max(x);
delay = length(m_short)-I;

if delay < 0
    delay = 0;
end

m_sync(1:(length(m)-delay),1) = m((delay+1):(length(m)));

m_no_guard = zeros(1, length(ttotal));

active = 0;
for symbreceived = 1:M+1;
%Removing guard
m_no_guard = m_sync((1+(length(tt)*(symbreceived-1))+guard_samples*(symbreceived-1)*active):
length(tt)*symbreceived+guard_samples*(symbreceived-1)*active);

if symbreceived == 1
    ref_chan = m_no_guard;
end

%Downshifting
r_tilde = m_no_guard.*exp(-i*2*pi*fc*tt');

%Lowpass filtering
[bb aa] = butter(13, 1/2);

r = filter(bb, aa, r_tilde);

%OFDM
info(1:length(r),symbreceived) = (1/FS).*fft(r);

%Removing Zero padding
info_1(1:(N/2),symbreceived) = info(1:freq_space_factor:(freq_space_factor*(N/2)), symbreceived);
info_1((N/2+1):N, symbreceived) = info((length(tt)-(freq_space_factor*(N/2)-1)):freq_space_factor:
length(tt), symbreceived);

active = 1;
```

```
end

info_1_dim = size(info_1);
data = zeros(N, M);
phase_shift_2 = zeros(N, M);

for rr = 2:info_1_dim(2);
    for ss = 1:info_1_dim(1);

    %Finding the phase difference
    phase_shift_2(ss,rr-1) = info_1(ss,rr)*conj(info_1(ss, rr-1));

    %Demodulating the differential coding
       if abs(imag(phase_shift_2(ss,rr-1)))>abs(real(phase_shift_2(ss,rr-1)));
            if imag(phase_shift_2(ss,rr-1))<0;     %-i
                data(ss,rr-1) = 1-i;
            elseif imag(phase_shift_2(ss,rr-1))>0; %i
                data(ss,rr-1) = -1+i;
            end
        elseif abs(imag(phase_shift_2(ss,rr-1)))<abs(real(phase_shift_2(ss,rr-1)));
            if real(phase_shift_2(ss,rr-1))<0;     %-1
                data(ss,rr-1) = 1+i;
            elseif real(phase_shift_2(ss,rr-1))>0; %1
                data(ss,rr-1) = -1-i;
            end
       end
    end
end

%Finding the QPSK errors
QPSK_errors = 0;
 for k = 1:M
     for l = 1:N
        if data(l,k)==Data_par(l,k)
        QPSK_errors = QPSK_errors;
        else
        QPSK_errors = QPSK_errors + 1;
        end
     end
end

DataIn = DataIn;
DataOut = QPSKdemodulator(data, N, M);

%If data_type is sound, a wav-file is created
if strcmp(data_type, 'sound');
   wavData = writeSound(DataOut, Fs_sound, bits, plots);
end

%Comparing the DataOut vector with DataIn vector
bit_errors = 0;
for j = 1:length(DataOut);
    if DataOut(j)==DataIn(j);
        bit_errors = bit_errors;
    else
        bit_errors = bit_errors + 1;
    end
end

clc;
%display in MATLAB
disp(['OFDM PARAMETERS:'])
disp(['Data type:                     ',data_type])
disp(['Symbol length:                 ',num2str(Tu),' s'])
disp(['Number of carriers:            ',num2str(N)])
disp(['Number of symbols:             ',num2str(M)])
disp(['Guard time:                    ',num2str(guard),' s'])
disp(['   '])

disp(['CHANNEL PARAMETERS:'])
disp(['Signal to noise ratio:         ' num2str(SNR) ' dB'])
disp(['Phase error:                   ' num2str(phase_error) ' rad'])
disp(['Impulseresponse:               ' num2str(impulseresponse) ' : 1 Enabled, 0 disabled'])
disp(['Delay:                         ' num2str(channel_delay) ' s'])
disp(['   '])

biterror = bit_errors/length(DataOut);
disp(['RESULTS:'])
disp(['Number of QPSK-symbol errors: ' num2str(QPSK_errors) '/' num2str(M*N)])
disp(['Number of bit errors:          ' num2str(bit_errors) '/' num2str(length(DataOut))])
disp(['BER:                           ' num2str(biterror)])
disp(['   '])

save('ref_chan.mat', 'ref_chan')

if plots == 1

%Plot of the baseband spectrum

figure();
subplot(211);
plot(f/1000, abs(fft(r, FS))/FS);
title('Spectrum of the received signal r(t)');
axis([0 192 0 1.5]);
xlabel('Frequency - kHz');
ylabel('Amplitude');
subplot(212);
pwelch(r,[],[],[],FS);

%Plot of the received phase shifts before decision
figure();
plot(phase_shift_2,'LineStyle','none', 'Marker','x', 'MarkerSize',10);
title(['Phase difference - SNR ', num2str(SNR), ' dB']);
axis([-0.5 0.5 -0.5 0.5]);
grid on;
xlabel('Real');
ylabel('Imag');

%Plot of the first 12 received QPSK Symbols
figure();
subplot(211);
stem(real(data(1:12)));
title('The real part of the data-vector');
axis([0 12 -1.5 1.5]);
set(gca,'ytick',[-1 1]);
xlabel('Symbol number');
ylabel('Value');
subplot(212);
stem(imag(data(1:12)));
title('The imaginary part of the data-vector');
axis([0 12 -1.5 1.5]);
set(gca,'ytick',[-1 1]);
xlabel('Symbol number');
ylabel('Value');
```

```
%Plot of the first 24 received bits and the soundOut
if strcmp(data_type, 'sound');
figure();
subplot(211)
stem(DataOut(1:24));
title('The DataOut-vector');
axis([0 24 -0.5 1.5]);
set(gca,'ytick',[0 1]);
xlabel('Bit number');
ylabel('Bit value');
subplot(212)
t = 0:1/Fs_sound:(length(wavData)-1)/Fs_sound;
plot(t, wavData);
title('The received wave-file')
axis([0 0.35 -0.6 0.6]);
xlabel('Time - Seconds')
ylabel('Amplitude')
else
figure();
stem(DataOut(1:24));
title('The DataOut-vector');
axis([0 24 -0.5 1.5]);
set(gca,'ytick',[0 1]);
xlabel('Bit number');
ylabel('Bit value');
end
end
```

QPSKdemodulator.m

```
%The function used to retrieve bits from the QPSK-symbols

function [a_data] = QPSKdemodulator(data, N, M)

dim = size(data);
a_data = zeros(2*N*M,1);
p = 2;

%Retrieving bits from the QPSK-symbol
for j = 1:1:dim(2);
    for k = 1:1:dim(1)

    if data(k,j) == 1 + i;
        a_data(p-1)=1;
        a_data(p)=1;
    elseif data(k,j) == -1 + i;
        a_data(p-1)=0;
        a_data(p)=1;
    elseif  data(k,j) == 1 - i;
        a_data(p-1)=1;
        a_data(p)=0;
    elseif data(k,j) == -1 - i;
        a_data(p-1)=0;
        a_data(p)=0;
    end
    p = p + 2;
    end
end
```

writeSound.m

```
%The function used to convert the bitstream into a wave-file

function [DataOut] = writeSound(bit_vector, Fs, bits, plots)

%Creating samples of 8 bits
dim = length(bit_vector);
teller2 = 1;
for h = 1:dim/bits;
    for g = 1:bits;

        u(h,g) = num2str(bit_vector(teller2));

        teller2 = teller2 + 1;
    end
end

%Binary to decimal
l = bin2dec(u);
DataOut = ((l-(2^(bits-1)))/(2^(bits-1))).';

%Creating soundOut.wav
wavwrite(DataOut, 'soundOut.wav');
```

channelMod.m

```
%The function used to simulate the channel

function sn_delay = channelMod(channel_delay, SNR, phase_error, impulseresponse)

load OFDMspace.mat;

delay_samples = round(channel_delay/T);

ttotal= 0:T:(Tu*(M+1)+M*T+T*delay_samples+(M)*guard); %Total time including delay

%CHANNEL MODELING
%phase error
theta = zeros(1,length(ttotal));

if phase_error > 0
%Linear phase error
theta = 0:(phase_error)/(length(s)):(phase_error);
end

s_phase_error = s.*exp(i.*-theta(1:length(s)));
s_phase_error_real = real(s_phase_error);

%Adding white gausian noise
S_0 = mean(s_phase_error_real.^2);

if SNR ~= -1
sigma = S_0/(10^(SNR/10));
else
sigma = 0;
SNR = inf;
end
```

```
AWGN = sigma*randn(size(s_phase_error_real));

sn = s_phase_error_real + AWGN;

%Adding delay
sn_delay = zeros(1,length(sn)+delay_samples);
sn_delay((delay_samples+1):length(sn_delay)) = sn;

%Multipaths
if impulseresponse == 1

    impulselength = 20e-3;
    timpulse = 0:T:impulselength;

    impresp = zeros(1, length(timpulse));

    impresp(1) = 1;
    impresp(200) = 0.6;
    impresp(400) = 0.4;
    impresp(600) = 0.3;
    impresp(800) = 0.25;

    s_conv = conv(sn_delay, impresp);
    t_conv = 0:T:(impulselength+ttotal(length(ttotal)));

    %Creating wave file
    MAX_SN = max(abs(s_conv));
    wavwrite(s_conv/MAX_SN, FS, 'signal');
    REF = ref/MAX_SN;

else
    %Creating wave file

    MAX_SN = max(abs(sn));
    wavwrite(sn_delay/MAX_SN, FS, 'signal');
    REF = ref/MAX_SN;
end

save('MAX_SN.mat', 'MAX_SN', 'SNR', 'phase_error', 'impulseresponse', 'channel_delay');

if plots ==1

%Plots

%Plot of the frequency error
figure()
plot(ttotal, theta(1:length(ttotal)))
title(['Phase error as a function of time']);
axis([0 5.2 0 10]);
set(gca,'ytick',[0 3.1415 6.2831],  'YTickLabel',{'0 ','pi ','2pi '});
xlabel('Time - Seconds');
ylabel('Phase error');

%Plot of the frequency spectrum of the received signal
figure();
subplot(211);
plot(f/1000, abs(fft(sn, FS))/FS);
title(['Spectrum of the received signal']);
axis([35 41 0 3]);
xlabel('Frequency - kHz');
ylabel('Amplitude');
```

```
subplot(212);
pwelch(sn,[],[],[],FS);
axis([35 41 -50 20]);

%Plot of the transmitted signal
figure();
subplot(211)
plot(ttotal, sn_delay);
title(['The transmitted signal']);
axis([0 5.2 -600 600]);
xlabel('Time - Seconds');
ylabel('Amplitude');
subplot(212)
plot(ttotal, sn_delay);
axis([1.748 1.76 -400 400]);
xlabel('Time - Seconds');
ylabel('Amplitude');

if impulseresponse == 1
%Plot of the impulseresponse of the channel
figure()
stem(timpulse, impresp);
title(['Channel impulse response']);
xlabel('Time - Seconds');
ylabel('Amplitude');

%Plot of the received signal
figure()
subplot(211)
plot(ttotal, s);
title(['The transmitted signal']);
axis([2.18 2.21 -200 200]);
xlabel('Time - Seconds');
ylabel('Amplitude');
subplot(212)
plot(t_conv, s_conv)
title(['The received signal']);
axis([2.18 2.21 -200 200]);
xlabel('Time - Seconds');
ylabel('Amplitude');
end
end
```

run.m

```
%The main file used during the simulations

clc;
clear all;
close all;

symbollength = 200e-3;                    %Length of the symbols
bandwidth = 2560;                         %Bandwidth in Hz
numberofcarriers = 512;                   %Number of carriers
numberofsymbols = 20;                     %Number of OFDM symbols
guard_time = 40e-3;                       %Guard_time in seconds
plots = 1;                                %Plots figures if value is set to 1
channel_delay = 0;                        %Delay in seconds
phase_error = 0;                          %Phase error in radians
signal_to_noise_ratio = -1;               %The chosen SNR in dB for the channel,
 -1 removes the noise
impulseresponse = 0;                       %1: enable channel impulse response

OFDMtransmitter(symbollength, bandwidth, guard_time, 'random', numberofcarriers,
numberofsymbols, plots);

channelMod(channel_delay, signal_to_noise_ratio, phase_error, impulseresponse);

OFDMreceiver;
```