
OFDM synchronization for the
MIMO testbed

Bachelor Assignment

Bram Ton

University of Twente
Department of Electrical Engineering,
Mathematics & Computer Science (EEMCS)
Signals & Systems Group (SAS)
P.O. Box 217
7500 AE Enschede
The Netherlands

Report Number: SAS03-10
Report Date: 21st April 2010
Period of Work: 02/05/2009 – 28/04/2010
Supervisors: Dr. ir. R. Schiphorst
Ir. X. Shao

List of Figures

2.1	Ideal OFDM transmitter block diagram [6]	4
2.2	IFFT input mapping [7]	5
2.3	OFDM frame structure [7]	5
3.1	Subcarrier frequency allocation [7]	8
3.2	Autocorrelation of a short preamble with 20dB of noise	10
3.3	Auto and cross correlation of a short preamble with 20dB of noise	12
3.4	Principle of Wang's method, SNR=20dB	13
3.5	Residual carrier frequency offset [2]	15
4.1	Correct frame detection region	20
4.2	Comparison of several frame detection methods with no channel model applied	20
4.3	Comparison of several frame detection methods with channel model applied	21
4.4	Comparison of the three different methods of coarse position detection, no channel model applied	22
4.5	Comparison of the three different methods of coarse position detection, with channel model applied	23
4.6	Fine peak estimation with and without channel model	24
4.7	Constellation diagram of an OFDM symbol, before and after channel equalization	25
4.8	Fine and coarse detections without channel model	27
4.9	Fine and coarse detections with channel model	27
4.10	Bit error rates with and without the channel model applied	28
5.1	Baseband setup	30
5.2	Setup using mixers	31
5.3	Real world setup	31
5.4	Magnitude of a received QPSK modulated signal	32
5.5	Constellation diagram of a QPSK modulated signal	33

5.6	Constellation diagram of a QAM modulated signal	34
5.7	Mean phase of the pilot signals	34

Contents

List of figures	ii
Table of Contents	iv
1 Introduction	1
1.1 Aims of this research	1
1.2 Research method	1
1.3 Overview of the following chapters	2
2 Background information	3
2.1 OFDM	3
2.2 802.11a standard	5
2.2.1 Preamble	5
3 Analysis of an OFDM system	7
3.1 Transmitter tasks	7
3.1.1 Data generation & modulation	7
3.1.2 Frames	8
3.1.3 Pilots	8
3.2 Receiver tasks	9
3.2.1 Frame detection	9
3.2.2 Time synchronization	11
3.2.3 Carrier frequency offset	13
3.2.4 Channel estimation and equalization	14
3.2.5 Residual carrier frequency offset	15
4 Simulations	17
4.1 Transmitter software blocks	17
4.1.1 Preamble generation	17
4.1.2 Frame generation	18
4.1.3 Carrier frequency offset	19
4.2 Receiver software blocks	19

4.2.1	Frame detection	19
4.2.2	Coarse detection	21
4.2.3	Fine detection	23
4.2.4	Carrier frequency offset compensation	24
4.2.5	Channel estimation and equalization	25
4.2.6	Demodulation	26
4.3	Integration	26
4.3.1	Bit error rates	26
5	Test setup	29
5.1	Testbed description	29
5.2	Test method	29
5.2.1	Baseband	30
5.2.2	Up and down	31
5.2.3	Air	31
5.3	Test results	31
6	Conclusions and recommendations	37
6.1	Conclusions	37
6.2	Recommendations	37
	Bibliography	40

Introduction

In the modern world both the mobility of people is increasing and their demand for digital information. To concede to these demands an efficient modulation scheme is needed which has high data rates and has a good performance in urban environments. Such a modulation scheme is Orthogonal Frequency Division Multiplexing (OFDM). The urban environment causes a distorted transmission channel between receiver and transmitter, which can be easily equalized when using the OFDM modulation scheme. OFDM is already used for the well known Wireless Local Area Network (WLAN) standard and the Digital Video Broadcasting standard. Besides channel equalization another important aspect of the transmission is the synchronization done by the receiver. It can even be stated that without synchronization between transmitter and receiver there would be no communication possible.

1.1 Aims of this research

This research will focus on how synchronization can be achieved and how the implementation of it can be accomplished. Several real world interferences which hamper the synchronization will be looked at and how these interferences can be dealt with. The final goal will be to make a software implementation which is able to demodulate the data received by the Multiple-Input Multiple-Output (MIMO) testbed. The MIMO testbed has been developed by the Signals and Systems group of the University of Twente to test various algorithms in real world office situations.

1.2 Research method

In the first stage of the research the principals of OFDM are investigated and simulated. The simulations will provide insight of the methods and will determine the right parameters. Some of the components of OFDM which are developed and tested are:

- Preamble generation
- Frame generation
- Carrier frequency offset
- Frame detection
- Coarse detection
- Fine detection
- Carrier frequency offset compensation
- Channel estimation and equalization

In the second stage of the research the data generated in the simulation will be tested on the MIMO testbed and analysed.

1.3 Overview of the following chapters

The second chapter will give the reader some background information on OFDM and the 802.11a standard, which is commonly used for wireless LAN. Some principles of this standard are used in this research. The next chapter will give an analysis of the methods used for synchronization, channel equalization and dealing with other real world interferences, such as carrier frequency offset. The fourth chapter will describe how the simulation of several methods mentioned in the previous chapter can be performed. This chapter will also present the results of the simulations carried out in this research . The last chapter will describe the method used for the real world tests and the results obtained will be presented.

Background information

This chapter provides a short introduction to OFDM and the 802.11a standard. This standard is commonly known as WLAN and provides data speeds up to 54 Mbps. The first section of this chapter gives a brief overview of OFDM. This section is followed by some background information on the 802.11a standard. Some principles of this standard are used in the simulations.

2.1 OFDM

OFDM is a modulation technique that uses multiple carriers which are orthogonal to each other. The number of carriers can be very large, for example the Digital Video Broadcasting - Terrestrial (DVB-T) standard uses 6816 subcarriers (for the 8k mode) [5]. The data being transmitted consists of a single stream of bits, called the bitstream. The bitstream being transmitted is first modulated using a digital modulation scheme such as Binary Phase Shift Keying (BPSK), Quadrature Phase Shift Keying (QPSK) or Quadrature Amplitude Modulation (QAM). After modulation the data is divided into N parallel streams, with N being the number of subcarriers. Each parallel stream is modulated on a subcarrier. After this, all N subcarriers are summed together and upconverted to the carrier frequency using a frequency mixer. Note that in some real life systems not all carriers are used to transmit data. Some subcarriers are used as pilots for channel estimation and phase compensation.

Each baseband OFDM symbol can be described by the following relation:

$$s(t) = \begin{cases} \sum_{k=1}^N d_k e^{j2\pi k \Delta f (t-t_s)} & t_s \leq t \leq t_s + T_s \\ 0 & \text{elsewhere} \end{cases} \quad (2.1)$$

In this relation the following symbols are used:

N Number of subcarriers

t_s Symbol start time

T_s Symbol duration

Δf Subcarrier spacing

d_i Complex data symbol

The process of generating N subcarriers and summing them all up is not feasible in a practical system, especially when using a large number of subcarriers. The solution is to use an Inverse Fast Fourier Transform (IFFT). The IFFT maps the frequency domain onto the time domain. The modulated N parallel streams can be seen as the frequency domain representation. These are mapped to the time domain by an IFFT consisting of N points. An IFFT can be implemented efficiently for powers of two. If N is not a power of two it can be zero padded to make it a power of two. A block scheme of an ideal OFDM transmitter can be seen in figure 2.1.

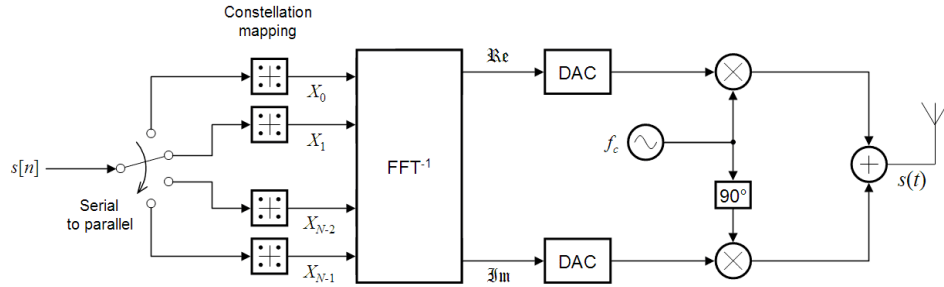


Figure 2.1: Ideal OFDM transmitter block diagram [6]

In this figure it can be seen that the bitstream is first converted to parallel and then modulated. The IFFT produces real and imaginary numbers. These numbers are converted to the analog domain by the Digital to Analog Converters (DAC). After this the signals are mixed with the carrier frequency. Note that the lower path is mixed using a carrier frequency which has a phase offset of 90 degrees. The upper path is called the in-phase component and the lower path is called the quadrature path. Finally both signals are added together and transmitted.

2.2 802.11a standard

The digital system implementation and the simulations use some principles of the IEEE 802.11a standard. The 802.11a standard uses 64 subcarriers and has a bandwidth of 20 MHz. This bandwidth is divided amongst the 64 subcarriers resulting in a Δf of 312.5 kHz. Each data OFDM symbol has a symbol time of $1/\Delta f$ in the case of 802.11a this results into a symbol time of $3.2 \mu s$. Of these 64 carriers only 52 are used, 48 are used for data and four are used for pilot carriers. Because an IFFT can be implemented efficiently for powers of two the 64 (2^8) samples are used to do the IFFT. The 52 samples are mapped to the IFFT input in a specific way. This is to overcome DC offset in the time domain signal. The input mapping can be seen in figure 2.2.

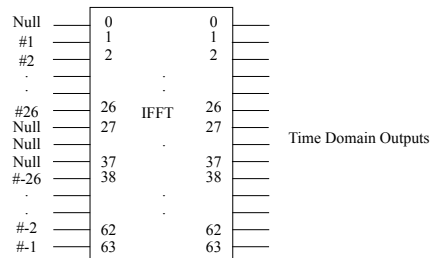


Figure 2.2: IFFT input mapping [7]

2.2.1 Preamble

Each frame being send consists of a preamble and data. The preamble is used for several things such as channel estimation, timing offset and frequency offset estimation. A 802.11a frame can be seen in figure 2.3.

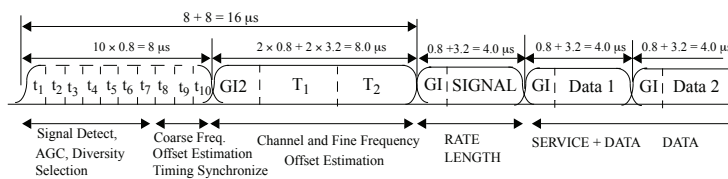


Figure 2.3: OFDM frame structure [7]

The first 160 symbols are called the short preamble and consists of ten identical blocks of sixteen symbols. Following the short preamble is the long preamble, which also consists of 160 symbols in total. It contains two identical blocks of 64 symbols each and a cyclic prefix of 32 symbols. The signal field which can also be seen in the figure is not used in the digital

system implementation. The data OFDM symbols consist of 80 symbols, of which 64 symbols represent the time domain version of the OFDM symbol and the remaining 16 symbols represent the cyclic prefix.

The preambles are constructed of symbol sequences which are 52 symbols of length and are defined in the 802.11a standard. The time domain representations of the preambles are given by their IFFT. The preamble is first mapped to a sequence which has a length of 64. The position at zero is set to zero and the values at the centre of the sequence are set to zero. This is to avoid a DC offset in the resulting signal. The power of both sequences have to be normalized. This is done by multiplying the short preamble with $64/\sqrt{24}$ and the long preamble with $64/\sqrt{52}$.

Analysis of an OFDM system

This chapter will describe the parts in the OFDM system which will be simulated. The OFDM system is divided into two parts, a transmitter part and a receiver part. The transmitter part will focus on how the signals which are transmitted are constructed. The receiver part will focus on how the data can be recovered again. Later on in Chapter 4 a description is given of a software implementation of an OFDM system. In this chapter the also the results of the simulations will be presented and a few notes are made about the implementation of the OFDM system.

3.1 Transmitter tasks

This section will describe the tasks which are done by the transmitter. The first subsection describes how the data is created and modulated. This is followed by a subsection describing how a frame is constructed and how the generated signals are modified to resemble a real world signal.

3.1.1 Data generation & modulation

The transmitted data is randomly generated. The modulation technique which will be used during most parts of the simulation is BPSK. This modulation technique is chosen because it is easy to implement. BPSK modulation maps a binary one to plus one and maps a binary zero to minus one. When simulating actual transmission and reception QPSK modulation will be used. When doing tests on the testbed QPSK and QAM with sixteen constellation points will be used. This will also be the modulation type when tests are done on the testbed. QPSK and QAM are used because in real world situations BPSK is not used often as it only sends one bit is modulated at a time. QPSK modulation maps a set of two bits to a single point in the constellation. No error correction will be used during the simulations as this is not part of the research question and is difficult to implement.

3.1.2 Frames

The frames used in the simulation start with a gap of 64 symbols. This number was arbitrarily chosen. The gap is used to test the frame detection and the symbol timing offset estimation. Following this gap is the preamble which has a total length of 320 symbols. The exact details about the preamble can be found in Section 2.2.1. Following this preamble are the OFDM data symbols. Each data symbol will consist of 80 OFDM symbols. These are 16 symbols for the guard interval and 64 symbols for the data. Of these 64 symbols only 52 actually contain data.

The simulation must also add noise to the frames and optionally apply the channel model. The channel model used in the simulations is the "Channel Model A" which was originally for the HIPERLAN/2 standard [8]. The channel model randomly attenuates or amplifies the subcarriers.

3.1.3 Pilots

In each OFDM data symbol four pilots are added. These pilots are BPSK modulated and are required to remove any residual carry frequency offset. When considering an OFDM symbol with an index ranging from -26 to 26, the pilots are inserted at positions -21, -7, 7 and 21 the values corresponding to these indices are [1, 1, 1, -1]. The pilot and data positions can be seen in Figure 3.1. Note that the spacing between pilots is fourteen samples. In the 802.11a standard the polarity of the pilot is cyclically changed to increase the robustness of the system [7]. In the simulations the polarities will not be cyclically changed to keep matters simple.

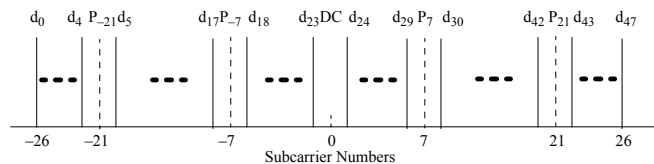


Figure 3.1: Subcarrier frequency allocation [7]

3.2 Receiver tasks

This section will describe the tasks which are done by the receiver in order to retrieve the original data. Some of these tasks are:

1. Frame detection
2. Synchronization
3. Carrier frequency offset compensation
4. Channel estimation & equalization
5. Residual carrier frequency offset compensation
6. Demodulate data

Each of these tasks will be looked at in the following subsections, except the demodulation part. Demodulation will not be looked at because it is not the main topic of the research.

3.2.1 Frame detection

The first thing the receiver has to do is check whether a frame is being received or not. The most simple way of doing this is by power detection. Power detection will look at the instantaneous power of the received signal and compare it to a predefined threshold. If the power exceeds the threshold a frame is detected. This method has some downsides, especially in noisy environments where outliers can occur easily. The frame detection must perform equally well in different signal-to-noise ratios. The above mentioned frame detection method will not be part of the simulations.

Two better alternatives for frame detection are discussed in the following two subsections. These are power detection using a window and correlation detections. During simulations of the frame detection algorithms it was clear the two mentioned algorithms did not work well enough so a third method was introduced. This new method is described last.

For the coarse timing offset detection to be correctly determined at least the last three symbols of the short preamble are needed. This is because the coarse detection will make use of the autocorrelation of the short preamble.

Frame detection will fail if the detector starts detecting in the middle of a frame. This error will be detected in the next phase of the demodulation, the coarse estimation. After such an error has been detected a possibility is to move ahead in time by a couple of samples and start with the frame detection again.

Power detection

Instead of looking at the power of the received signal at a single moment in time and comparing this to a threshold one could also look at a moving averaged power in time. By using an average outliers caused by noisy environments are cancelled out. The average power is calculated by using a window which is moved along the received samples. The window will move one sample at the time and calculates the average power inside the window. A frame is detected if the power average exceeds a specified threshold.

An even better way of determining a frame is to check whether the average power has increased over time. A very small chance exists that a frame is falsely detected. This can occur when the noise power increases in a successive manner. The chance that this will happen is so small that it will be neglected.

Correlation detection

Due to the repetitive structure of the preamble, which is described in Section 2.2, a frame can easily be detected by performing an autocorrelation. Each frame starts with a preamble which starts with ten exactly the same symbol sequences. A figure showing an autocorrelation of the short preamble is shown in Figure 3.2. The detection of the frame can be done by comparing

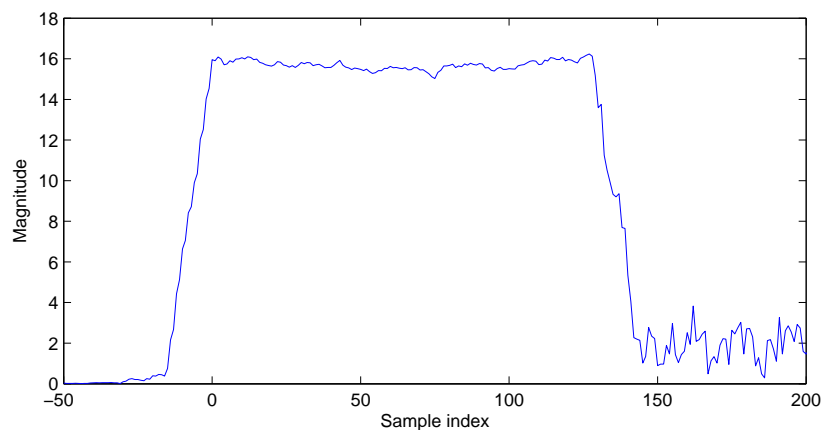


Figure 3.2: Autocorrelation of a short preamble with 20dB of noise

the value of the autocorrelation to a threshold. If a frame is being received the autocorrelation value will increase. In the same manner as the power detection a moving average can be used. As can be seen in the figure the autocorrelation increases over time. A threshold can be set to define the minimum amount of successive increases before a signal is classified as a frame.

Noise power detection

This method first determines the average noise power of the signal using a window. The noise power value is used later on as a threshold. A window is moved over the signal and the average is calculated. The window does not move at one sample a time, but moves a window length at a time. This is because the position of the frame does not have to be very precise. The average power is compared to the previously calculated threshold. If the power is above the threshold the signal is classified as a frame.

3.2.2 Time synchronization

After a frame has been detected the exact start of the payload must be determined. To determine this start different methods exist which are all based on the principle of correlation. The synchronization is commonly split up into two parts, the first part does a rough estimation of the symbol timing and the second part does a precise estimation of the symbol timing.

Coarse estimation

The coarse estimate is usually done by an autocorrelation of the received signal. The autocorrelation formula can be seen in Equation 3.1.

$$P(d) = \sum_{m=0}^{L-1} r_{d+m}^* r_{d+m+L} \quad (3.1)$$

The sampled received signal is represented by r . Note that d is the time index in a window of $2L$ samples. This autocorrelation can also be written as an iterative formula [9]. This is convenient because less calculations have to be done. The iterative formula can be seen in Equation 3.2.

$$P(d+1) = P(d) + (r_{d+L}^* r_{d+2L}) - (r_d^* r_{d+L}) \quad (3.2)$$

In our case L will be sixteen as this is the symbol length of a symbol in the short preamble. Two autocorrelation based methods for the rough estimate will be discussed.

The first method determines the maximum of the autocorrelation function and determines at which sample index the autocorrelation has decreased by an given percentage from the maximum. A typical autocorrelation output can be seen in Figure 3.3. In this figure also the cross-correlation output is shown. This will be used later on when determining the precise symbol offset. The peaks of the cross-correlation correspond to the *start* of each symbol in the short preamble. The rough estimate point should be somewhere between the last and second last peak of the cross-correlation. This is because the fine symbol offset detection must determine the exact position of the last peak. The problem of this method is that finding a corresponding

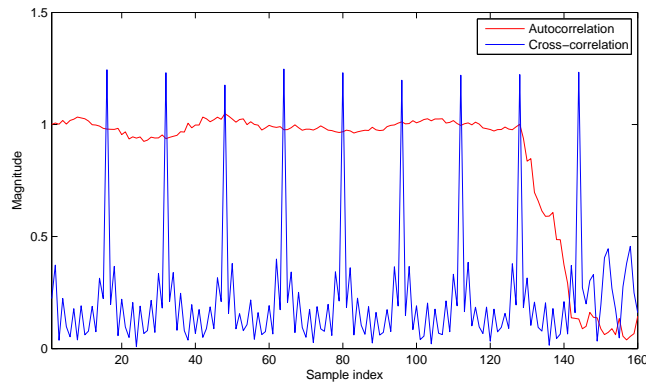


Figure 3.3: Auto and cross correlation of a short preamble with 20dB of noise

index for a arbitrary value in an array is difficult. A possible implementation would search the smallest difference of the absolute error. The absolute error is calculated by subtracting the arbitrary value from each value in the array and taking the absolute. The corresponding index can now be found by searching the smallest value. The index which is found may not correspond to the value which is wanted but to noise which lies closer to the arbitrary value. When the frame is detected too early the index might also correspond to a value which lies in the rising slope of the autocorrelation. A possible solution for this particular problem is to only search for an index value which lies past the index of the maximum value. This solution has not been tested in the simulations.

Another way to estimate the rough symbol time is by using two autocorrelations where one is delayed by 32 sampling instances. These two are subtracted from each other creating a peak. This peak can easily be detected using a `max()` function, the position of this peak is used as rough estimation. This method is described by Wang [12] and the principle of this method can be seen in Figure 3.4.

Fine estimation

After the rough estimation the cross-correlation is used to determine a more precise point for the symbol timing. Cross-correlation makes use of the fact that the receiver knows what the transmitter has transmitted. Each symbol in the short preamble gives a distinct peak in the cross-correlation, this can be seen in Figure 3.3.

The last peak in the cross-correlation figure corresponds to the start of the last symbol in the short preamble. Knowing this and the position of the coarse point, only a small part of the signal has to be cross-correlated

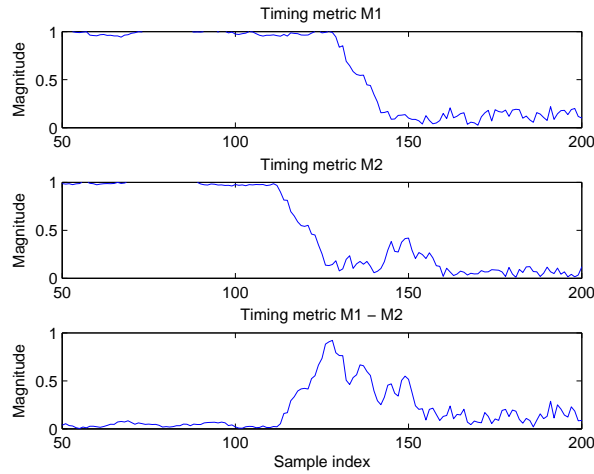


Figure 3.4: Principle of Wang's method, SNR=20dB

to find the last peak. In the resulting output the last peak can easily be detected by using a `max()` function.

3.2.3 Carrier frequency offset

Carrier frequency offset (CFO) occurs when the local oscillators of transmitter and receiver are not well synchronized to each other. Carrier frequency offset can also occur due to the Doppler effect. A mismatch in carrier frequencies causes inter-carrier interference (ICI). If this happens the orthogonality between subcarriers is lost. The effect of CFO can be simulated by multiplying the time domain signal with a factor $\exp(j2\pi\Delta ft)$ this will add a phase rotation [3].

This phase rotation is later estimated at the receiver and by using the following relation:

$$\phi = 2\pi T \Delta f \quad (3.3)$$

The carrier frequency offset can be determined. The T mentioned in the relation is the symbol time. The frequency offset can now be determined as follows:

$$\widehat{\Delta f} = \frac{\hat{\phi}}{2\pi T} \quad (3.4)$$

A phase can only be resolved if it lies in the range of $[-\pi, \pi]$, this corresponds to a frequency range of $[-1/2T, 1/2T]$. For example in the case of the 802.11a standard where the short preamble symbols have a symbol time of $0.8 \mu s$ the maximum frequency offset is 625 kHz.

The determination of the carrier frequency offset is divided in two steps. First the coarse estimation is done using the short preamble symbols. The

signal is then corrected using the coarse frequency offset. After this a fine estimation is made using the symbols of the long preamble and the signal is corrected. The correction of the carrier frequency offset can be done by multiplying the time domain signal starting from the long preamble onwards with $\exp(-j\phi t)$. The time t mentioned in the relation starts at zero at the long preamble start. Note that there will be a difference in t between transmitter and receiver. This will cause a constant phase for each sample in the time domain signal. This phase is compensated for when the channel is equalized.

3.2.4 Channel estimation and equalization

After a frame has been detected, the symbol offset has been determined and the carrier frequency offset is corrected the channel has to be estimated. The channel estimation is done using the long preambles. Assuming the time domain signal after Fast Fourier Transform (FFT) looks like:

$$Y(k) = C(k)X(k) + Z(k) \quad (3.5)$$

Where C is the channel, X is the transmitted signal and Z is the noise, a simple channel estimation can be done by dividing the received signal by the known preamble [4]. The channel estimation $\hat{C}(k)$ will become:

$$\hat{C}(k) = \frac{Y(k)}{X(k)} \quad (3.6)$$

This channel estimation method is known as the zero forcing algorithm and can be used to compensate the received signal by dividing the received signal with the channel estimation.

3.2.5 Residual carrier frequency offset

Even after the carrier frequency offset has been compensated for and after the channel is equalized still some residual carrier frequency offset may exist. The residual CFO contained in the received signal may very well be time-varying and thus needs to be continuously tracked [3]. The residual CFO causes phase distortions of the OFDM symbol in the frequency domain. Assuming that the CFO is constant for the received signal the residual CFO will increase proportional to the symbols, this can be seen in Figure 3.5 where θ_s is the initial phase and θ_d is the phase increase per symbol [2]. Note that the residual CFO can also decrease proportional to the symbols.

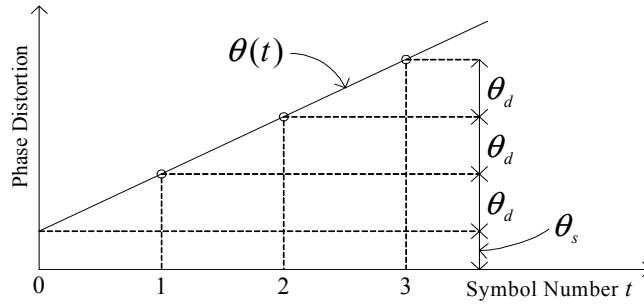


Figure 3.5: Residual carrier frequency offset [2]

The total residual phase of symbol number t would be:

$$\theta = \theta_s + t\theta_d \quad (3.7)$$

To compensate for the residual CFO the frequency domain signal is multiplied with $\exp(-j\theta)$.

Simulations

Before testing the digital system implementation on the MIMO testbed simulations are done first. These simulations are done in order to test the principal of OFDM and to optimize parameters. This chapter gives a explanation about how the simulations are done. The results of the simulations will not be presented in a separate chapter but are presented and discussed on the go. All simulations are done using the high-level technical computing language Matlab.

The simulation of the OFDM system is divided into two parts, the receiver and the transmitter respectively. At first all blocks will be individually tested, to avoid dependencies on other blocks. This is to ensure that if a block is modified not all dependant blocks have to be tested again. If all blocks work they will be tested as a whole. Also at first only noise is added and later the channel model will be used in the simulations.

This chapter is divided into three sections, the first section will deal with the transmitter simulations. This is followed by a section about the receiver simulations. And finally the last section describes the chain of transmitter and receiver combined.

4.1 Transmitter software blocks

This section will describe some tasks which are typically done by the transmitter side. The tasks include the generation of preambles and the assembling of frames. The tasks are described in the coming two subsections.

4.1.1 Preamble generation

Before the frames can be assembled the preamble has to be constructed. This is done using the sequences mentioned in the 802.11a standard. Only the creation of the short preamble will be described here, the long preamble is done in exactly the same manner. A piece of code describing the genera-

```

preamble = [];
symbol = zeros(1,64);
symbol(2:27) = Tshort(28:53);
symbol(39:64) = Tshort(1:26);
% Power normalisation
symbol = (64/sqrt(2*12))*symbol;
symbol = ifft(symbol);
% ten times 16 samples
preamble = [symbol symbol symbol(1:32)];

```

Listing 4.1: Short preamble generation

```

% Random bit sequence
data = randint(Nframes*SymbolsPerFrame, 52, 2);
% BPSK modulation
data = 2*data -1;

```

Listing 4.2: BPSK modulation

tion of the short preamble can be seen in Listing 4.1. The variable *Tshort* contains the symbols of the short preamble.

4.1.2 Frame generation

The preambles described in the previous section are appended to the frames. Each frame will consist of a 'gap', the preambles and the data symbols. The data is BPSK modulated, which is very easy to implement in Matlab as can be seen in Listing 4.2. Note that each row represents a data symbol to be transmitted and has a length of 52 OFDM symbols. A single frame can contain more than one data symbol, this is taken into account when generating the frame. Later on QPSK will be used when actually transmitting and demodulating data. The QPSK modulation is done by using the Matlab Modem Object. This object can take care of the modulation and demodulation of several types. A short snippet of code describing the QPSK modulation can be seen in Listing 4.3. Note that there is an extra factor

```

% Random bit sequence
data = randint(Nframes*SymbolsPerFrame*52*2, 1, 2);
% QPSK modulation
h = modem.qammod('M', 4, 'InputType', 'Bit',
    'SymbolOrder', 'Gray');
data = modulate(h, data);
data = reshape(data, 52, Nframes*SymbolsPerFrame)';

```

Listing 4.3: QPSK modulation

two in the random data generation, this is because the modulation works as a kind of compression. After modulation there are 52 symbols left over. It is needed to reshape the data because when modulating a multi-dimensional array Matlab assumes the columns are individual channels and the rows are timestamps.

During simulation average white Gaussian noise (AWGN) is added to the frames. Optionally also a channel model is applied to the frames. This is done by convolving the frame with a row from the 'Channel Model A' data set. During simulation no pilots are added to the signal, but when testing on the testbed pilots are added to the signal. As a consequence of this the simulations will use 52 data bits whilst the testbed measurements will use 48 data bits. This leaves four positions open which will be used as pilots.

4.1.3 Carrier frequency offset

After the addition of noise the last thing which is added to the signal is frequency offset. This is done by multiplying the time domain signal with $\exp(\phi t)$ as can be seen in Listing 4.4.

```
phase = 2*(pi/180);  
t = 0:length(frame)-1;  
frame = frame.*exp(i*phase.*t);
```

Listing 4.4: Adding carrier frequency offset

In this listing a phase of two degrees is successively added to the signal, thus the first sample does not have a phase offset, the second sample has a phase offset of two degrees, the third sample has a phase offset of four degrees etc. When assuming a symbol time of $3.2 \mu s$ this result into a Δf of 1.7 kHz.

4.2 Receiver software blocks

This section will describe some tasks which are typically done by the receiver side. These tasks include the timing offset detection, channel and frequency correction and the demodulation.

4.2.1 Frame detection

To compare all three frame detection algorithms mentioned in Section 3.2.1 a test was done using different signal-to-noise ratios. The signal-to-noise ratio was varied between -20 and 20 dB. For each signal-to-noise ratio a thousand frames were tested to obtain the percentage of correctly detected frames. For each of the three frame detection methods a window size of sixteen was used. A frame was considered correctly detected if it was detected before the third last symbol of the short preamble and forty samples before the start of the frame. The region previously described can be seen in Figure 4.1. Both

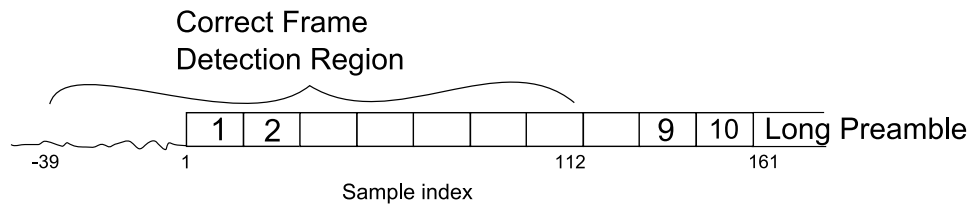


Figure 4.1: Correct frame detection region

the power detection method and the autocorrelation detection method need sixteen successive increases in their output before the signal is classified as a frame. The results of all three methods when no channel model is used can be seen in Figure 4.2. The frame detection ratio mentioned in the

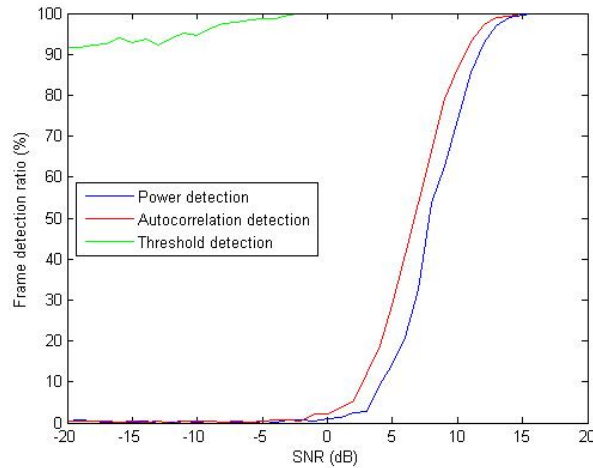


Figure 4.2: Comparison of several frame detection methods with no channel model applied

figure is the percentage of detected frames to the total amount of frames. It can be seen from the figure that the noise power method has a far better performance compared to the other two detection methods. In Figure 4.3 the results of all three frame detection methods are shown when using a channel model. It can be seen that the performance slightly decreases. In a real life situation it is not known whether the received signal consists of frames, noise, or a combination of both. This may lead to situations where a received signal is falsely classified as a frame. To test how many frames are falsely detected a thousand frames consisting of only noise are tested to see if they are detected as frame. The noise level was varied between -20 dB and 20 dB and no channel model was used. The results showed that the threshold detection method was the worst, on average roughly 92%

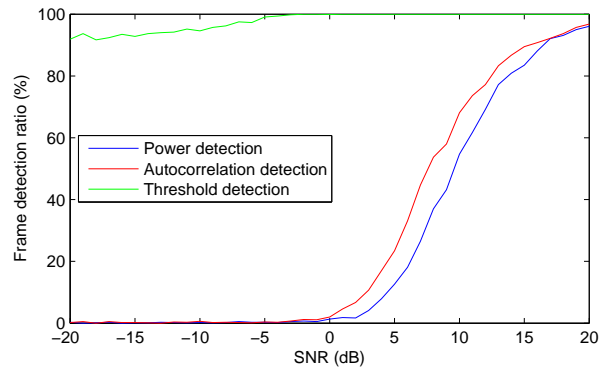


Figure 4.3: Comparison of several frame detection methods with channel model applied

```
P = abs(P);
poi = percentage*max(P); % Our point of interest
[rte , pos] = min(abs(P - poi));
```

Listing 4.5: Sample index determination

of noise frames were detected as an actual frame. The other two methods performed much better. Using the correlation based detection only 0.31% of the frames were falsely detected and the power detection method performed a little better were only 0.17% of all frames were falsely detected.

The high number of falsely detected frames for the noise power threshold method can be explained. When only comparing the averages of noise to each other, the chance is very high that their is an average which is slightly higher. This will result to a falsely detected frame. An improvement would be not to compare the averages to each other but to move the threshold a few standard deviations to the right.

4.2.2 Coarse detection

All three methods for coarse timing estimation, which are mentioned in Section 3.2.2, are compared with each other for values of signal-to-noise ratio between 0 dB and 10 dB. For each signal-to-noise ratio a thousand frames are tested. After determining the value at which the autocorrelation has decreased by the given percentage the corresponding sample index has to be determined. This is done by subtracting the value from the rest of the signal and looking where the difference is the smallest. To elaborate this the corresponding code snippet is given in Listing 4.5. In this listing P represents the autocorrelation of the signal. The poi is the percentage of the maximum value. The corresponding sample index is now found by subtracting the

```

P = abs(P);
% Our point of interest
poi = percentage*max(P);
rte = abs(P - poi);
tmp = rte < 0.5*mean(rte);
tmp = find(tmp);
if isempty(tmp)
    [rte, pos] = min(rte);
else
    pos = tmp(1);
end

```

Listing 4.6: New sample index determination

autocorrelation output and the *poi* and looking at the minimum value of the result.

As can be seen the above mentioned method looks at the minimum value after subtraction. There might be a possibility that the minimum value does not correspond to the wanted value but to noise. The improved method solves this issue by looking at the first occurrence where the error value is smaller than half the average error. If this is not found it will just use the old method to find the sample index. The principle can be seen in Listing 4.6. The results can be seen in Figure 4.4. The figure shows the percentage of frames where the coarse position has been correctly detected. From the figure it can be seen that the improved max method

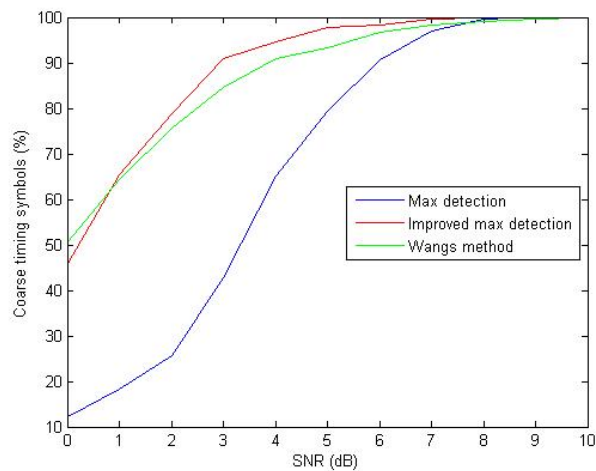


Figure 4.4: Comparison of the three different methods of coarse position detection, no channel model applied

and the method by Wang almost perform equally well. For both methods

based on the maximum principle a correct detection was registered if the detected position lays in the second last symbol of the short preamble. For the method of Wang a correct detection was registered if the symbol time was off at maximum eight samples from the start of the second last symbol of the short preamble.

The same test as above was repeated, but this time using a channel model. The results using can be seen in Figure 4.5. It can be seen that

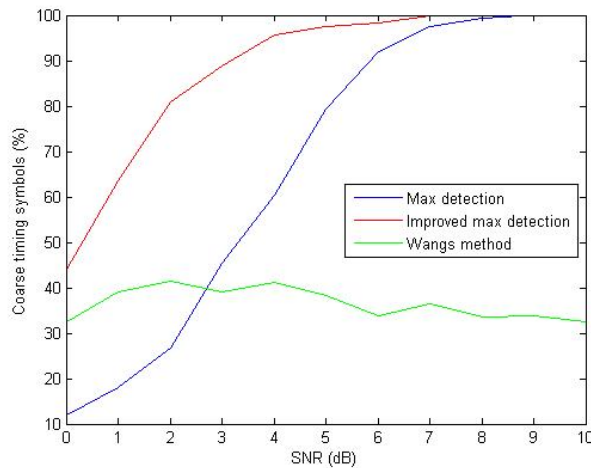


Figure 4.5: Comparison of the three different methods of coarse position detection, with channel model applied

the Wang method performs significantly worse when compared to the other methods. The method which will be used for the rest of the simulations will be the improved max function.

To test whether false detections occur of the coarse positions frames consisting of different values of noise are tested. For each value of noise in the range of 0 dB to 10 dB ten thousand frames are tested without the channel model. The improved max function does not falsely detect any positions. The old max function falsely detects coarse positions in 9% of the cases. The method of Wang falsely detects the positions in 11% of the cases.

4.2.3 Fine detection

The fine detection is the last step and determines the symbol time offset. The fine peak algorithm needs a coarse position so that it knows where to look. For the coarse position in the simulation an arbitrary position of 137 was chosen. In the simulation ten thousand frames were used per different value of signal-to-noise ratio. The signal-to-noise ratios were varied between

```

% Coarse CFO determination , using short preambles
P = sum(conj(buffer(pos:pos+31)).*buffer(pos+32:pos+63));
phase = angle(P)/32;

% Compensate for coarse CFO
t = 0:length(buffer)-1;
buffer = buffer.*exp(-i*phase.*t);

```

Listing 4.7: Carrier frequency offset estimation using short preambles

-20 dB and 10 dB. Both a test with channel model and one without channel model was conducted, the results can be seen in Figure 4.6. A correct detection was only registered if the deviation was at maximum four samples compared to the correct position. The correct position is the start of the last symbol in the short preamble.

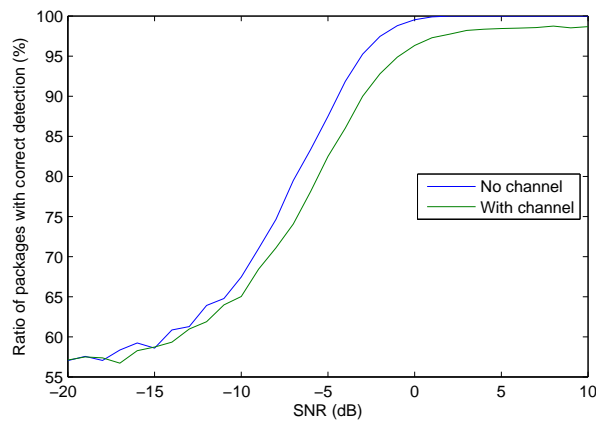


Figure 4.6: Fine peak estimation with and without channel model

4.2.4 Carrier frequency offset compensation

Before the channel is estimated the received signal is corrected for the carrier frequency offset. The compensation is done using the phase of the signal. The phase is estimated using the autocorrelation of both short and long preambles. The total phase correction is commonly done in two steps. The first step determines a coarse phase offset using four symbols of the short preamble. This process can be seen in Listing 4.7. Note that in the listing the phase is divided by 32 because the average phase is calculated by the autocorrelation. After the coarse phase has been determined the signal is first corrected using this phase. This is done by multiplying the time domain signal again with the inverse of the found phase as can be seen in Listing 4.4.

```
LP_t = buffer(LP_start:LP_start+63);  
tmp = fft(LP_t);  
LP_f(1:26) = tmp(39:64);  
LP_f(27:52) = tmp(2:27);  
H = LP_f./Tlong;
```

Listing 4.8: Channel estimation

The fine phase estimation is done in the same manner as the coarse phase estimation, but this time the two long preamble symbols are used.

4.2.5 Channel estimation and equalization

After the carrier frequency offset correction the channel estimation is done by dividing the frequency domain representation of the received long preamble with the actual frequency domain representation of the long preamble. The code listing can be seen in Listing 4.8. In the code listing LP_t represents the time domain representation of the long preamble and LP_f represents the frequency domain representation. This is divided by the actual frequency domain representation which is represented by $Tlong$ in the code to obtain the channel estimation. In Figure 4.7 a constellation diagram of a single OFDM data symbol is plotted before and after channel equalization. QPSK modulation was used here and the signal-to-noise ration was 20 dB. The image on the left shows the constellation diagram of an uncompensated OFDM symbol. The image on the right shows the constellation after channel compensation and the four quadrants can clearly be seen.

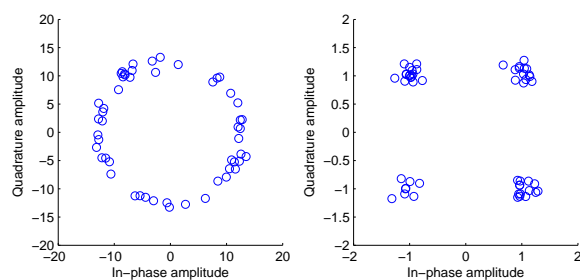


Figure 4.7: Constellation diagram of an OFDM symbol, before and after channel equalization

4.2.6 Demodulation

The demodulation of the signal is taken care of by the Matlab modem object. The demodulation object must match the one which is used in the modulation phase. To get a global idea about how the demodulation object can be used Listing 4.9 is provided.

```
demodObject = modem.qamdemod('M', 4, 'SymbolOrder', 'gray',  
    'OutputType', 'bit');  
z = demodulate(demodObject, symbol');
```

Listing 4.9: QPSK demodulation object

The *symbol* mentioned in the code is a single OFDM symbol which has the guard interval removed and has been channel corrected. Later on the carrier frequency offset will be introduced to the signal and will also be compensated for.

4.3 Integration

This chapter describes the integration of all above mentioned simulation pieces. During testing it became clear that the frame detection was not working properly. The frames were detected too early, resulting in a malfunction of the coarse detection algorithm. This malfunction is caused by the fact that the rising slope of the autocorrelation is present. When looking at the percentage of the maximum value of the autocorrelation a value may be detected which lies in the rising slope region of the autocorrelation. The rising slope can clearly be seen in Figure 3.2. To avoid this problem the frame position is manually set and everything works remarkably well as can be seen in Figure 4.8. To generate this figure the signal-to-noise ratio was varied between -5 dB and 15 dB, for each noise level a thousand frames were tested. Note that the results used for this figure did not use the channel model yet. The figure shows both the amount of coarse peak detections, and the amount of fine peak detections. It can be seen that both lines are nearly equal meaning that if the coarse peak detection fails also the fine peak detection fails. When the channel model is enabled the results can be seen in Figure 4.9. It is remarkable to see that there is a difference between the two lines, meaning that if the coarse detection fails there is a possibility that the fine detection does succeed.

4.3.1 Bit error rates

An interesting measurement in the communications world is the bit error ratio (BER). The BER is the number of bit errors divided by the total number of bits transmitted. This test was also done in the simulations, one with no channel model and one with a channel model. To obtain the results

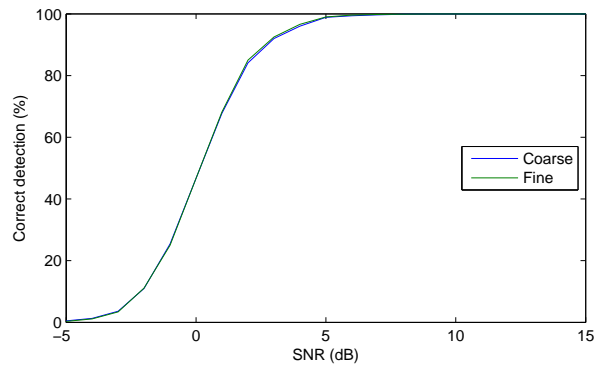


Figure 4.8: Fine and coarse detections without channel model

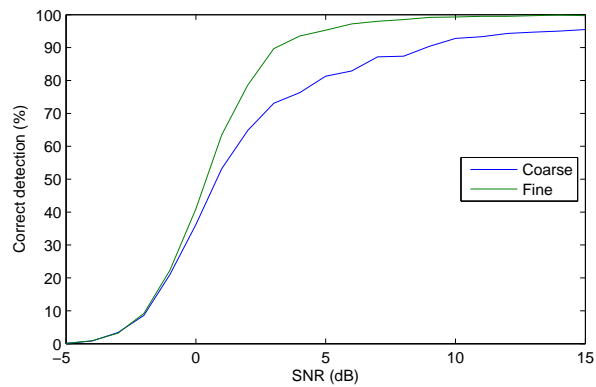


Figure 4.9: Fine and coarse detections with channel model

the signal-to-noise ratio was varied between -5 dB and 15 dB and for each signal-to-noise level a thousand frames with ten symbols each were tested. This time also QPSK modulation is used, and the frame start is manually set, as the frame detection method does not work well. The results can be seen in Figure 4.10.

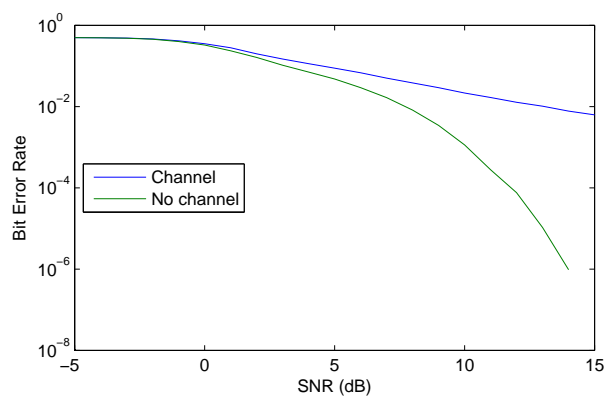


Figure 4.10: Bit error rates with and without the channel model applied

Test setup

After the simulations the OFDM principal is tested in real life on the Multiple-Input Multiple-Output (MIMO) testbed. The real-time MIMO testbed enables researchers to test algorithms in a real-world office situation, to change environment conditions and add or remove processing power in a flexible fashion [10].

5.1 Testbed description

The testbed is able to transmit and receive four channels simultaneously and has a modularized setup. The setup consist of two trolleys which act as the transmitter and receiver. The transmitting trolley contains a digital to analog converter, a up-converter, a RF amplifier and an Uninterruptible Power Supply (UPS) so that also over large distances can be transmitted. The receiver trolley contains a down-converter and a analog to digital converter. Both transmitter and receiver have a data buffer. The transmitter will continuously transmit the contents of the buffer and at the receiver side the data can be captured into the buffer and can be read into the computer as a raw stream file.

In the following sections the blocks in the test setup images correspond to the physical blocks present on the trolleys. This done to keep things simple, but might also be somewhat misleading. For example, the down-converter also contains a low pass filter.

5.2 Test method

The baseband signals are generated by the Matlab code and have to be first converted to an IT++ file. This IT++ file is later converted to a raw stream file by a custom piece of C++ software. The raw stream file is then loaded into the data buffer of the transmitter. The received data is also in the raw stream format and is first converted to the IT++ format which then can be

converted back to a Matlab readable format. The data conversion between the IT++ format and the Matlab variable format is done by two Matlab scripts called *itload.m* and *itsave.m* respectively. These files are provided by the IT++ project [1]. The experiments will be done in a progressive manner. This means that first a signal is transmitted at baseband this is followed by a signal which is up converted and then down converted again and finally if everything works the signal is transmitted through the air. Each of the tests will be done using three different modulation scheme which have an increasing difficulty to detect. The three modulation schemes are, in order of difficulty, BPSK, QPSK and Quadrature Amplitude Modulation using sixteen symbols (QAM16). Each of the experiments will be described briefly in the following subsection. The blocks which are depicted in the figures correspond to the physical blocks which are present on the MIMO testbed.

5.2.1 Baseband

First only the baseband transmission is done, this means the signal has not been mixed up to the carrier frequency and transmitter and receiver are still connected by a wire. The system setup can be seen in Figure 5.1. The

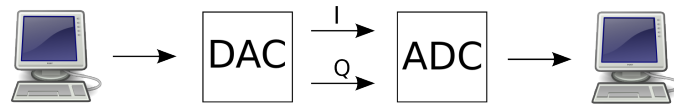


Figure 5.1: Baseband setup

computer on the left will act as the transmitter and will place the data in the transmission buffer. The buffer is read out by the digital-to-analog converter (DAC) and produces an in-phase (I) and a quadrature (Q) baseband signal per channel. The output of the DAC is attached to the I and Q inputs of the analog-to-digital converter, which will write the received data to another buffer. This buffer can be read out with the computer on the receiver side.

5.2.2 Up and down

The next step is to up and down convert the signal in the transmission path. The transmitter and receiver are still connected to each other using a cable. The setup can be seen in Figure 5.2. The blocks with an arrow

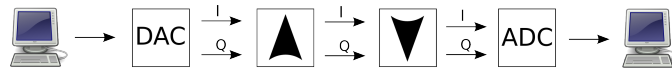


Figure 5.2: Setup using mixers

symbol represent the mixers. These mix the baseband signal with the carrier frequency.

5.2.3 Air

The last and final step will transmit the data through the air. The setup can be seen in Figure 5.3. Note that there is a power amplifier (PA) attached to

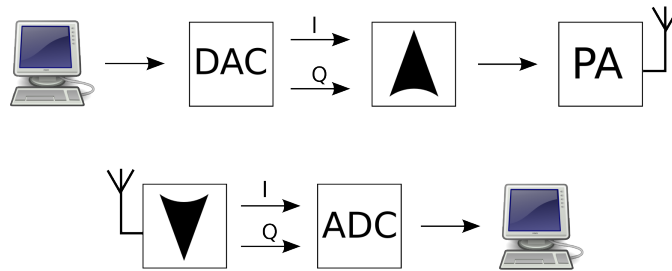


Figure 5.3: Real world setup

the DAC, this is because the DAC can not produce enough power by itself for transmission.

5.3 Test results

The test results which were obtained from the testbed were not as expected. The problem might be the testbed or the baseband data which is generated by the Matlab script. As time was a critical issue the testbed results of Xiaoying Shao were used instead. These results only contained QPSK and QAM16 modulated data.

Each measurement contained seven packets which consisted of 16 frames, this means that for each measurement 112 frames are demodulated. The number of OFDM symbols per frame for QPSK modulated data was 128 and for QAM16 modulated data was 64. This was to ensure that the same amount of bits were transmitted for each modulation type. Each transmitted packet contained the same data. The OFDM data symbols contained

pilots, this was needed to compensate for the residual carrier frequency offset. Especially the QAM16 modulated signals were effected by this effect. The tests which used a carrier frequency the frequency was set to 2.3 GHz. This frequency was chosen to avoid any interferences with existing WLAN transmitters which use a carrier frequency of 2.4 GHz. An external local oscillator was used for both the ADC and the DAC as the internal oscillator does not work well. The Automatic Gain Control (AGC) of the receiver was also adjusted by hand as it did not function properly.

The initial idea was to transmit a single frame and do the frame detection by hand as the frame detection algorithms which were tested did not work properly. Because the data of Xiaoying contained seven packets with 16 frames it would be very laborious to do the frame detection by hand for each frame. A very simple frame detection method was used which compared the instantaneous power of the received signal to predefined threshold. This method could be used because the signal-to-noise ratio (SNR) of the received signals was very high. The magnitude of a received QPSK modulated signal corresponding to the start of a frame can be seen in Figure 5.4. From the figure it can clearly be seen that the SNR is high and that the received signal shows some repetition.

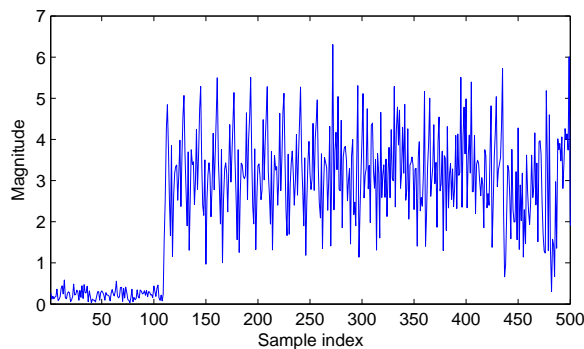


Figure 5.4: Magnitude of a received QPSK modulated signal

The measurement data consisted of seven sets. The most important parameters of these sets are summarized in Table 5.1. This table shows the type of modulation used, the estimated signal-to-noise ratio which is calculated from the mean energy of the signal (E_s) and the variation (σ^2). The dynamic range of the channel is also mentioned and is depicted by the \mathcal{D} symbol.

A constellation diagram of the first frame of a QPSK modulated signal can be seen in Figure 5.5. The constellation points do not show any overlap with other constellation points, this means the data can be recovered error free.

no.	modulation	SNR (dB)	E_s	σ^2	\mathcal{D} (dB)
1	QPSK	22.5	1.63	0.0091	8
2	QPSK	20.5	1.56	0.0138	35
3	QPSK	21.0	1.04	0.0082	22
4	QPSK	19.9	1.33	0.0136	17
5	QAM	22.5	1.33	0.0075	30
6	QAM	23.4	1.63	0.0076	5
7	QAM	22.7	1.53	0.0081	14

Table 5.1: Information about measurement data

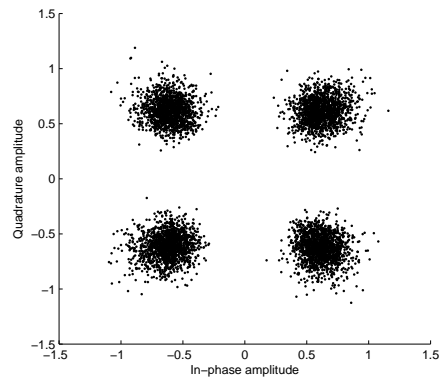


Figure 5.5: Constellation diagram of a QPSK modulated signal

A constellation diagram of the first frame of a QAM modulated signal can be seen in Figure 5.6. It can be seen that the complete constellation diagram is turned a bit to the left. This is caused by some constant carrier frequency offset which has not been removed properly.

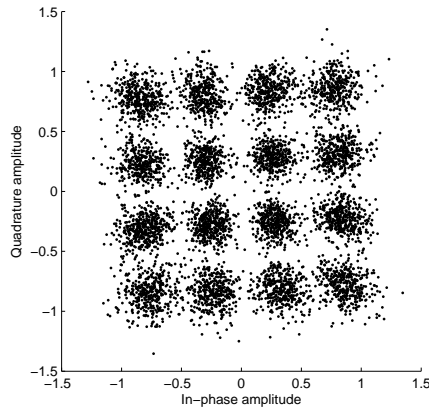


Figure 5.6: Constellation diagram of a QAM modulated signal

A plot of the mean phase of the pilots of the first five frames of the second measurement can be seen in Figure 5.7. The abrupt changes in phase correspond to the different frames.

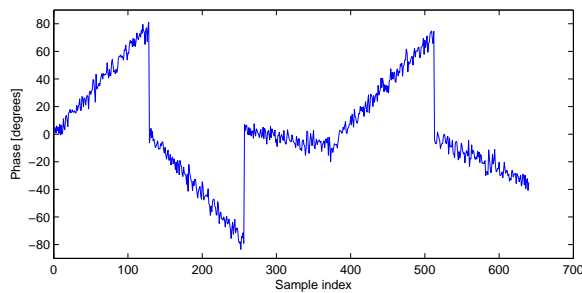


Figure 5.7: Mean phase of the pilot signals

To test the robustness of the developed Matlab scripts some artificial noise was added to the noise because the measurement data already has high signal-to-noise ratios. Three different levels of noise were added, 15 dB, 10 dB and 5 dB. Only three values were chosen because the Matlab demodulation script takes a long time to execute. For each of the measurements the bit error rate as a percentage was determined. The final results can be seen in Table 5.2. The first column in the table is the measurement, the number is the same as the one used in Table 5.1. The second column are the bit error rates expressed as a percentage for the measurements without artificial

no.	modulation	no noise	15 dB	10 dB	5 dB
1	QPSK	0,00	0,00	0,01	0,13
2	QPSK	1,60	1,80	2,16	3,43
3	QPSK	0,18	0,47	1,30	10,31
4	QPSK	0,16	0,24	0,48	4,75
5	QAM	3,05	4,08	5,77	10,69
6	QAM	1,16	1,53	2,37	4,92
7	QAM	2,06	2,88	4,73	9,47

Table 5.2: Bit error rates for various levels of artificial noise.

noise added. The following columns are the bit error rates, once again expressed as a percentage, for measurements with artificial noise added. The Matlab script which does the synchronization and demodulation takes about six minutes to run on a 2.0 GHz Intel Core Duo.

An interesting fact to notice is that measurements were QAM modulation was used the bit error rates are slightly higher when compared to the measurements were QPSK modulation was used. This is possibly caused by the fact that the signal-to-noise ratio of the QAM measurements are higher 5.1.

Conclusions and recommendations

6.1 Conclusions

The results from the simulation show that algorithms developed to carry out the different tasks in the OFDM system like the preamble and frame generation, frame detection and channel estimation and equalization are working. Some of the modules like the frame detection need some adjustment to improve the detection rate. The improved max detection method appears to be the most efficient method for detecting the coarse position.

The developed Matlab software has a modular setup such that new parts can easily be developed, tested and implemented. The software implementation which does the synchronization and demodulation of the received signals is working in real life situations. The bit error rates which have been achieved on the MIMO testbed are between 0 % and 3,05 % for signals with a signal-to-noise ratio of roughly 20 dB. Now it is proven that the signal received by the MIMO testbed can be synchronized and demodulated. Future development of various algorithms can be carried out easily on the MIMO testbed.

6.2 Recommendations

Some recommendations for future work are listed below.

- Eliminate the need to do file conversions manually, at the moment two file conversions are done before the measurement data can be imported into Matlab.
- Decrease the execution time of the Matlab script. The current Matlab script takes six minutes to execute at the moment for a single measurement.

- A frame detection algorithm which performs well under low signal-to-noise ratios and that is not dependant on a threshold.
- Extend the Matlab scripts such that they also work for the MIMO case.

Bibliography

- [1] IT++. <http://apps.sourceforge.net/wordpress/itpp>.
- [2] K. Akita, R. Sakata, and K. Sato. A phase compensation scheme using feedback control for ieee 802.11a receiver. In *Vehicular Technology Conference, 2004. VTC2004-Fall. 2004 IEEE 60th*, volume 7, pages 4789–4793 Vol. 7, Sept. 2004.
- [3] T. Chieh and P. Tsai. *OFDM Baseband Receiver Design for Wireless Communications*. John Wiley & Sons (Asia) Pte Ltd, 2007.
- [4] Y. Chiu, D. Markovic, H. Tang, and N. Zhang. *OFDM Receiver Design*. PhD thesis, Berkeley, 2000.
- [5] ETSI. Digital Video Broadcasting (DVB); Framing structure, channel coding and modulation for digital terrestrial television, 01 2001.
- [6] O. Filth. File:OFDM receiver ideal.png, 2009. http://en.wikipedia.org/wiki/File:OFDM_transmitter_ideal.png.
- [7] IEEE. 802.11a-1999 High-speed Physical Layer in the 5 GHz Band, 2008.
- [8] J. Medbo and P. Schramm. Channel models for HIPERLAN/2 in different indoor scenarios. Technical Report 3ERI085B, ETSI/BRAN, Mar. 1998.
- [9] T.M. Schmidl and D.C. Cox. Robust frequency and timing synchronization for ofdm. *Communications, IEEE Transactions on*, 45(12):1613–1621, Dec. 1997.
- [10] X. Shao. The MIMO project, 2008. http://www.sas.el.utwente.nl/open/research/wireless_communication/MIMO.
- [11] R. van Nee and R. Prasad. *OFDM for Wireless Multimedia Communications*. Artech House, 2000.

- [12] K. Wang, M. Faulkner, J. Singh, and I. Tolochko. Timing Synchronization for 802.11a WLANs under Multipath Channels.