UNIVERSITÀ DEGLI STUDI DI PISA

FACOLTÀ DI INGEGNERIA DELLE TELECOMUNICAZIONI

TESI DI LAUREA SPECIALISTICA - CORSO DI SISTEMI DI TRASMISSIONE

Synchronization in UWB systems

CANDIDATO: Luca Cairone

RELATORI:
Ing. Marco Moretti
Prof. Umberto Mengali
Prof.dr.ir Alle-Jan van der Veen

Anno Accademico 2006/2007

"La teoria è quando si sa tutto e niente funziona. La pratica è quando tutto funziona e nessuno sa il perché. Noi abbiamo messo insieme la teoria e la pratica: non c'è niente che funzioni... e nessuno sa il perché!"

A. Einstein

Abstract

A technology called UWB (Ultra Wideband) has gained recent attention. Using UWB, the scarce spectrum resource could be shared by several users. By spreading the signal over a wide frequency range, the average power per Hertz will be very low. The signal is very noise-like and is claimed to be able to co-exist with existing systems and services. The FCC and ETSI are considering allowing UWB, in already allocated bands, under its current regulation. Some parties have raised objections that the aggregate effects of a large number of UWB devices may raise the noise floor considerably. There may be a risk of interference with existing systems. One version of UWB, previously also called Impulse Radio, has the potential of being implemented with CMOS technology. This could result in very inexpensive transceiver chips. Due to the extreme bandwidth used, exceptional properties have been claimed. UWB is also claimed to have good multipath resolution. These properties are very important for indoor geolocation.

This thesis is focused on one of the most interesting subjects of research for UWB technology: the synchronization. A synchronization algorithm is proposed, claimed able to solve the presence of Inter Frame Interference (IFI). After that, an algorithm to detect the presence of the signal is proposed. Everything is done in a simple way to keep the receiver complexity very low. The work was developed in the group Circuits and Systems, faculty of Electrical Engineering, Mathematics and Computer Science, at TU-Delft, since 16 of August till 15 February , under the supervision of Prof. dr. ir. Alle Jan van der Veen and the collaboration of Yiyin Wang, PhD student in the same group.

Contents

Chapter 1

Introduction

1.1 UWB technology

Ultra-wideband is a radio technology. It can be used at very low energy levels for short-range high-bandwidth communications by using a larger portion of the radio spectrum. This method is using pulse coded information with sharp carrier pulses at a bunch of center frequencies in logical connex. UWB has traditional applications in non cooperative radar imaging. Most recent applications target sensor data collection, precision locating and tracking applications. Ultra Wideband was traditionally accepted as pulse radio, but the FCC and ITU-R now define UWB in terms of a transmission from an antenna for which the emitted signal bandwidth exceeds the lesser of 500 MHz or 20% of the center frequency. Thus, pulse-based systems (wherein each transmitted pulse instantaneously occupies the UWB bandwidth, or an aggregation of at least 500 MHz worth of narrow band carriers, for example in orthogonal frequency division multiplexing (OFDM) fashion) can gain access to the UWB spectrum under the rules [1] [2]. Pulse repetition rates may be either low or very high. Pulse-based radars and imaging systems tend to use low repetition rates, typically in the range of 1 to 100 megapulses per second. On the other hand, communications systems favor high repetition rates, typically in the range of 1 to 2 giga-pulses per second, thus enabling short range gigabit-per-second communications systems. Each pulse in a pulse-based UWB system occupies the entire UWB bandwidth, thus reaping the benefits of relative immunity to multipath fading (but not to intersymbol interference), unlike carrier-based systems that are subject to both deep fades and intersymbol interference. A February 14, 2002 Report and Order by the FCC authorizes the unlicensed use of UWB in 3.1÷10.6 GHz.

The FCC power spectral density emission limit for UWB emitters operating in the UWB band is -41.3 dBm/MHz. This is the same limit that applies to unintentional emitters in the UWB band, the so called Part 15 limit (figure 1). However, the emission limit for UWB emitters can be significantly lower (as low as -75 dBm/MHz) in other segments of the spectrum. Due to the extremely low emission levels currently allowed by regulatory agencies, UWB systems tend to be short-range and indoors applications. However, due to the short duration of the UWB pulses, it is easier to engineer extremely high data rates, and data rate can be readily traded for range by simply aggregating pulse energy per data bit using either simple integration or by coding techniques. Conventional OFDM technology can also be used subject to the minimum bandwidth requirement of the regulations. High data rate UWB can enable wireless monitors, the efficient transfer of data from digital camcorders, wireless printing of digital pictures from a camera without the need for an intervening personal computer, and the transfer of files among cell phone handsets and other handheld devices like personal digital audio and video players.

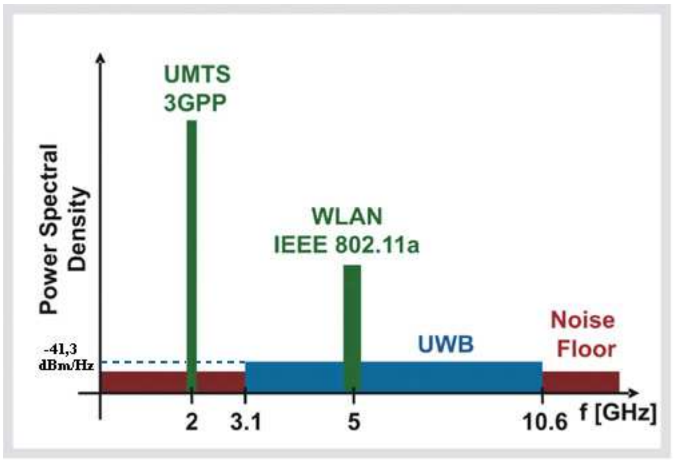

Figure 1.1: Electromagnetical spectrum occupied by UWB signals

1.2 Synchronization in UWB

In any communication system, the receiver needs to know the timing information of the received signal to accomplish demodulation. The subsystem of the receiver which performs the task of estimating this timing information is known as the synchronization stage. Synchronization is an especially difficult task in spread spectrum systems which employ spreading codes to distribute the transmitted signal energy over a wide bandwidth. The receiver needs to be precisely synchronized to the spreading code to be able to despread the received signal and proceed with demodulation. Timing acquisition is a particularly acute problem faced by UWB systems [3], as explained in the following. Short pulses and low duty cycle signaling employed in UWB systems place stringent timing requirements at the receiver for demodulation. The wide bandwidth results in a fine resolution of the timing uncertainty region, thereby imposing a large search space for the acquisition system. Moreover, the transmitted pulse can be distorted through the antennas and the channel, and hence the receiver may not have exact knowledge of the received pulse signal waveform. Typical UWB channels can be as long as 200 ns [4] [5], and can be characterized by dense multipath with thousands of components for some NLOS scenarios. The transmit-reference (TR) scheme first proposed for UWB in [6] [7] emerges as a realistic candidate that can effectively deal with these challenges. By transmitting pulses in pairs (or doublets) in which both pulses are distorted by the same channel, and using an autocorrelation receiver, the total energy of the channel is gathered to detect the signal without having to estimate individual channel multipath components. The simple delay (at the transmitter), correlation and integration operations (at the receiver) ease the timing synchronization requirements and reduce the transceiver's complexity.

In this thesis a TR-UWB Communication System is considered. Transmitted Reference UWB uses ultra-short information bearing pulses and thus promises high speed, high precision, resolved multipath and simpler receiver structures.

The same system was considered by Andreas Schranzhofer in his Master's Thesis [8] at TU Delft. He proposed to make synchronization by correlating the received samples with the code sequence known at the receiver. It's the traditional "matched filter". The limitation in [8] is that it assumes there is no Inter Symbol Interference (ISI) or Inter Frame Interference (IFI).

In [9] another acquisition scheme is proposed, in which, two sets of direct sequence code sequence are used to facilitate coarse timing and fine aligning. In this case, no IFI is assumed. A very complex algorithm is proposed in [10]. It proposes a blind synchronization method for TR-UWB systems. The matrix decomposition brings to a very high complexity. In the CAS group, a new algorithm is proposed, claimed able to solve the presence of IFI, with a very low complexity. The thesis focuses on the new proposed algorithm.

Mathematical notation

$\boldsymbol{v}$	boldface uncapitalized characters denote vectors
$\boldsymbol{M}$	boldface capitalized characters denote matrices
$\tilde{\boldsymbol{v}}, \tilde{\boldsymbol{M}}$	the operator ˜ means the FFT of the vector/matrix
x, A	italic characters denote scalars or complex numbers
$\lfloor x \rfloor$	denotes the first integer smaller than x
$\otimes$	denotes the Kronecker product of two matrices
$\odot$	denotes the Schur-Hadamard product of two matrices
$\star$	denotes the convolution product

Chapter 2

Data Model

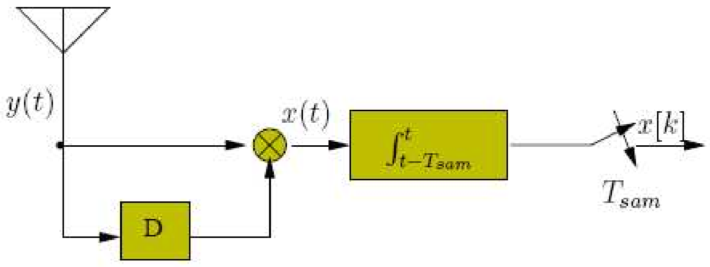

Figure 2.1: Autocorrelation receiver

This chapter presents the data model as described in [11]. Let's start introducing the model of a single frame, of duration T_f. Then we'll extend the model to multiple frames and multiple symbols.

2.1 Single frame

When a UWB pulse $g(t)$ is transmitted, it usually undergoes a distortion due to the UWB physical channel $h_p(t)$, supposed to be of finite length T_h. The signal is picked up by the antenna receiver, whose impulse response is $a(t)$. Consequently the expression of the received signal is:

$$h(t) = h_p(t) \star g(t) \star a(t) \tag{2.1}$$

$h(t)$ comprises the communication channel, the transmitted pulse and the antenna response. Actually, it hasn't been considered the bandpass filter present after the antenna receiver, but $a(t)$ could be seen also as the convolution between the antenna response and the bandpass filter response. Anyway, from now on, we'll consider $h(t)$ as the "composite" channel response. In TR-UWB systems, the signal is transmitted sending a pair of pulses per frame, called doublet. The first pulse of each doublet is fixed, called reference pulse, whereas the second pulse, sent after D seconds from the first, is the data pulse and its polarity s_0 contains the information: $s_0 \in \{-1, +1\}$. The expression of the signal due to one transmitted frame, after the antenna receiver and the bandpass filter is therefore:

$$y_0(t) = h(t) + s_0 \cdot h(t - D)$$

In figure 2.1 the receiver structure (leaving out the bandpass filter) is shown. As we can se, $y_0(t)$ undergoes a multiplication by a delayed version (by D seconds) of itself. After that, the result is integrated and dumped. The sampling period is T_{sam} and it usually is an oversampling, since P samples per frame are taken. P is called "oversampling factor": $T_{sam} = \frac{T_f}{P}$.
The expression of the signal at the multiplier's output is

$$\begin{aligned} x_0(t) &= y_0(t)y_0(t-D) \\ &= [h(t) + s_0 h(t-D)][h(t-D) + s_0 h(t-2D)] \\ &= [h(t)h(t-D) + h(t-D)h(t-2D)] + s_0[h^2(t-D) + h(t)h(t-2D)] \end{aligned}$$

Let's define the channel autocorrelation function as

$$R(\tau, n) = \int_{(n-1)T_{sam}}^{nT_{sam}} h(t)h(t-\tau)dt.$$

After the integration and the oversampling, we have the samples:

$$x_0[n] = [R(0, n - \frac{D}{T_{sam}}) + R(2D, n)]s + [R(D, n) + R(D, n - \frac{D}{T_{sam}})] \tag{2.2}$$

In equation (2.2) $R(0, n - \frac{D}{T_{sam}})$ contains the energy of the channel, in the time interval $[(n-1)T_{sam}, nT_{sam}]$. Let's consider a certain correlation length τ_0: as shown in [11], when $\tau_0 < D$ we can neglect the other terms of equation (2.2). Usually τ_0 is very small, often smaller than the delay D. Therefore we can ignore the terms $R(\tau, \ldots)$ with $\tau \epsilon \{D, 2D\}$. For the sample $x_0[n+1]$ the time interval involved is $[nT_{sam}, (n+1)T_{sam}]$: we can look at the oversampling process as a segmentation of the channel in "sub-channels", because $T_{sam} < T_f < T_h$. For each segment we have a channel autocorrelation function and a dominant term $R(0, \ldots)$ that contains the energy of the sub-channel. Since T_h is the length of the physical channel, we'll have P_h segments, where $P_h = \lfloor \frac{Th}{T_{sam}} \rfloor$. From now on, P_h samples will describe the original composite channel response:

$$h[n] = \int_{(n-1)T_{sam}}^{nT_{sam}} h^2(t)dt \qquad n = 1, \ldots, P_h \tag{2.3}$$

So, we can Define a TR-UWB "channel" vector:

$$\boldsymbol{h} = [h[1], \ldots, h[P_h]]^T \tag{2.4}$$

We can stack the discrete samples in a vector $\boldsymbol{x_0}$, obtaining:

$$\boldsymbol{x_0} = \boldsymbol{h} \cdot s_0 + noise \tag{2.5}$$

This model is a simple approximation for single frame. It reduces the complexity, even for the receiver algorithms and was shown that the approximation doesn't produce any considerable loss in the BER performance. It is due to the statistical properties of the UWB channels and to the nature of the UWB signal.
Using this model, it is simple now to derive data models for multiple frames.

2.2 Multiple frames

Let's consider now the transmission of N_f consecutive frames, each one with duration T_f and for each one the preceding data model is valid. Now, the data pulse of each doublet carries the information of a symbol: a data bit s_j modulates the polarity of the second pulse of the frame number j. We can now also introduce the presence of inter-frame interference (IFI), since the duration of each frame T_f is shorter than the channel length T_h.
We use a single delay for all the frames, that is D seconds, so that the receiver structure is the same as in figure 2.2. We have now new cross terms in the result of the integration and dumping, because there are more than one frame. They still are terms of the autocorrelation functions of the channel segments and they also can be ignored since the correlation length in these cross-terms are much longer than T_f. What we cannot ignore is the new matched terms that spread over some next frames, due to the fact that $T_h > T_f$. These other matched terms produce IFI. Let's define a channel matrix $\boldsymbol{H}$ to model the multiple frames case:

$$\boldsymbol{x} = \boldsymbol{H}\boldsymbol{s} + \boldsymbol{noise} \tag{2.6}$$

where $\boldsymbol{x}$ contains all the received samples, $\boldsymbol{s}$ is the unknown data vector

$$\boldsymbol{s} = [s_1, \ldots, s_{N_f}]^T$$

The channel matrix $\boldsymbol{H}$ has the structure shown in figure 2.2. The first thing to notice is the presence of shifted version of $\boldsymbol{h}$ defined in (2.4). Then, let's notice also the effect of IFI, looking at how many rows in $\boldsymbol{H}$ have more than one entry different by zero.

2.3 Effect of timing synchronization

In UWB communication systems, as already said, the pulses have very short duration. It makes the synchronization have stringent requirements. To solve the problem we work in the digital domain, elaborating the samples received. So, the analog part of the receiver can be kept data-independent.
Let's suppose to receive the data packet (consisting of multiple frames) with an offset G at the beginning, which means, in other words, that we are not synchronized.

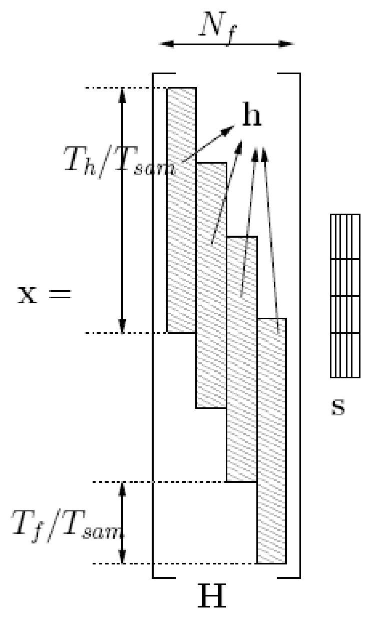

Figure 2.2: Data model for multiple frames

The expression of the offset is:

$$G = G'T_{sam} + g$$

where G' is an integer and g respects the condition: $0 < g < T_{sam}$. We can incorporate the integer G' in the data model putting G' rows at the top of the channel matrix $\boldsymbol{H}$ with all elements equal to zero. However we still have to include the offset fraction g and we can do it redefining the channel vector $\boldsymbol{h}$ as follows:

$$h[n] = \int_{(n-1)T_{sam}}^{nT_{sam}} h^2(t-g)dt, \quad n = 1, \ldots, P_h$$

Actually, we didn't any assumptions on the unknown channel vector $\boldsymbol{h}$, so, the model (2.6) is still valid. In the chapter 3, we'll explain the algorithm to estimate the delay G'. Now we continue considering $G' = 0$.

2.4 Multiple frames, Multiple symbols - Single User case

Let's go now to extend the preceding models to the multiple frames and multiple symbols case. Let's suppose to transmit a packet of N_s data symbols $\boldsymbol{s} = [s_1, \ldots, s_{N_s}]^T$, where each symbol $s_i \epsilon \{-1, +1\}$ is composed of N_f frames, whose duration is still T_f. In each frame, the pulses are separated by D seconds. Let's suppose then $c_{ij} \epsilon \{-1, +1\}, \quad j = 1, \ldots, N_f$ to be the known user code of the frame number j in the symbol number i. It means that the code can varie from frame to frame and from symbol to symbol. The structure of the receiver is still shown in figure 2.1 and the structure of the transmitted pulse sequence is shown in figure 2.3.

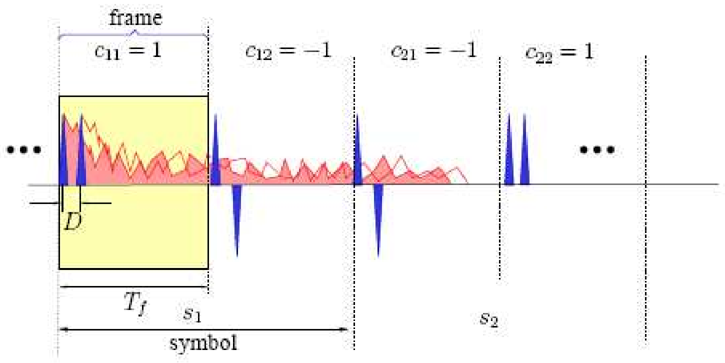

Figure 2.3: Pulse sequence structure

The expression of the signal after the antenna receiver and the pass-band filter, without noise, is:

$$y(t) = \sum_{i=1}^{N_s} \sum_{j=1}^{N_f} [h(t-((i-1)N_f+j-1)T_f)+s_i c_{ij} h(t-((i-1)N_f+j-1)T_f-D)] \quad (2.7)$$

where $\boldsymbol{c}_i = [c_{i1}, \ldots, c_{iN_f}]^T$ is the code vector for the $i-$th symbol s_i. Then we have the multiplication $x(t) = y(t)y(t-D)$, the integration and dumping with oversampling factor $P = T_f/T_{sam}$. As said in the section 2.1, The unmatched terms and the cross-terms can be neglected. The data model in (2.6) can be easily extended to include the code c_{ij}.

We still stack the samples $x[n] = \int_{(n-1)T_{sam}}^{nT_{sam}} x(t)dt, \quad n = 1, \ldots, (N_s N_f - 1)P + T_h/T_{sam}$ into a column vector $\boldsymbol{x}$, whose expression is now (see fig. 2.4)

$$\boldsymbol{x} = \boldsymbol{H} diag\{\boldsymbol{c}_1, \ldots, \boldsymbol{c}_{N_s}\}\boldsymbol{s} + \boldsymbol{noise} \tag{2.8}$$

where, $\boldsymbol{H}$ still have the same structure shown in fig. 2.2, and the '*diag*' operator puts the vectors $\boldsymbol{c}_1, \ldots, \boldsymbol{c}_{N_s}$ into a block diagonal matrix. We can also rewrite the

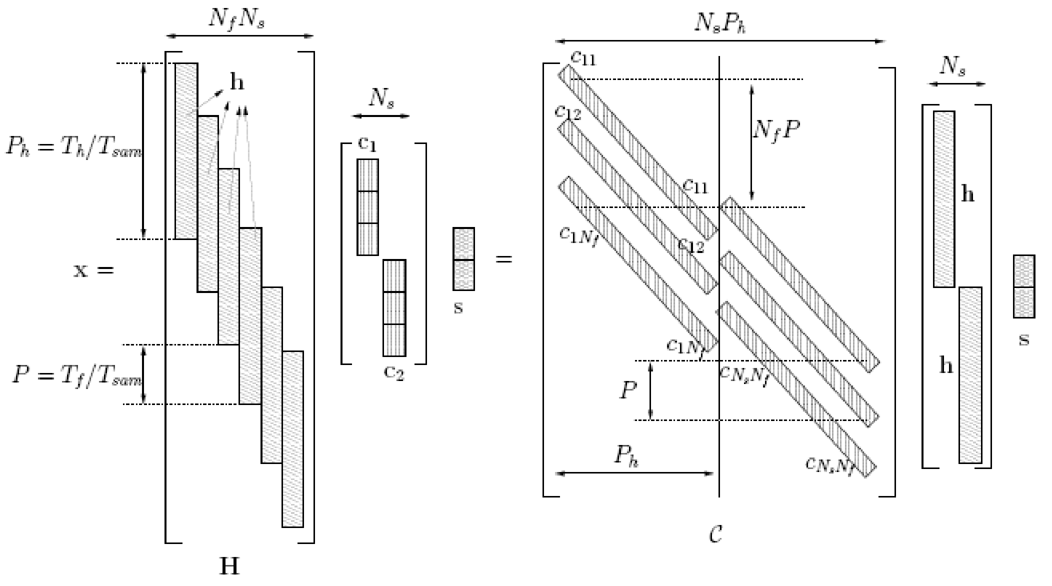

Figure 2.4: The data model for the single user case with no offset

data model in (3.1) as,

$$\boldsymbol{x} = C(\boldsymbol{I}_{N_s} \otimes \boldsymbol{h})\boldsymbol{s} + \boldsymbol{noise} \tag{2.9}$$

where $\otimes$ denotes the Kronecker product and C is the code matrix of size

$$((N_f N_s - 1)T_f + T_h)/T_{sam} \times (T_h N_s)/T_{sam}$$

The structure of $\boldsymbol{x}$ is shown in figure 2.4, where we can see also the structure of the matrix C.
As we said at the end of section 2.3, in the following chapter, the data model will be modified to introduce the the offset and illustrate the algorithm to solve the synchronization.

Chapter 3

The synchronization algorithm

A synchronization algorithm is proposed in this chapter. It was developed by Yiyin Wang, PhD student in the group of Circuits and Systems at TU-Delft. The main advantage of this algorithm is to solve the IFI problem in the synchronization through the code sequence deconvolution done in frequency domain. The traditional code match filter can't handle IFI. It requires the frame length to be long enough to let the channel die out before the next frame is transmitted. The proposed algorithm has lower complexity than the code match filter. The novel algorithm facilitates higher data rate communication and reduces the acquisition time. Yiyin Wang is now improving the algorithm as we can see in [12].

3.1 Data model

The scenario here is to solve the synchronization problem for single user with single delay. The received N_s symbols at the antenna output without noise can be modeled as in equation (2.7). The proposed training sequence has the following property:

$$s_k = s_{k+1} = s_{k+2} = \ldots = s_{k+N_s}$$

As already said, the duration of the channel T_h is much longer than the frame time T_f. This assumption introduces Inter Frame Interference. To avoid Inter Symbol Interference we insert a guard interval between two following symbols. The guard interval is equal to $T_h - T_f$. The integration and the oversampling (remember figure 2.1 produces the samples $x[n]$:

$$x[n] = \int_{(n-1)T_{sam}}^{nT_{sam}} y(t)y(t-D), \quad n = 1, \dots, N_s L_s \tag{3.1}$$

where $L_s = N_f P + P_h - P$ is the symbol length in terms of number of samples (see figure 3.1).

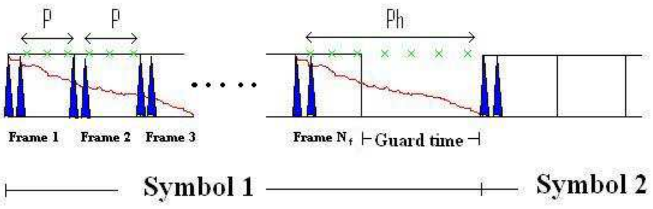

Figure 3.1: Samples per symbol period

$x[n]$ is stacked into a column vector as described in the chapter 2:

$$\boldsymbol{x} = C(\boldsymbol{I} \otimes \boldsymbol{h})\boldsymbol{s} = \boldsymbol{H} diag\{\boldsymbol{c}_1, \dots, \boldsymbol{c}_{N_s}\}\boldsymbol{s} \tag{3.2}$$

Here C is slightly different from C in chapter 2, there is no overlap between symbols in C, due to the guard interval inserted. See figure 3.2:

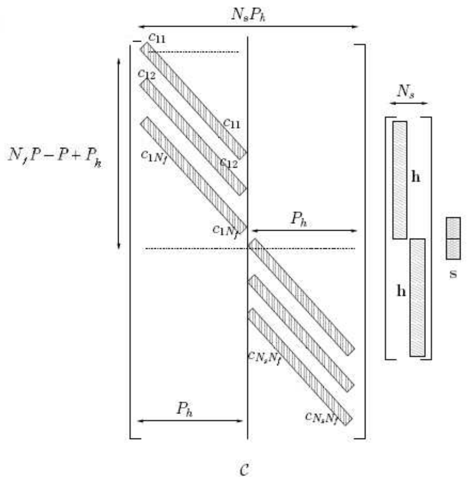

Figure 3.2: The data model

$\boldsymbol{h}$ is still the channel energy vector defined in (2.3). $\boldsymbol{s}$ is the symbol vector, whereas

$\boldsymbol{H}$ in the equation (3.2) is a $N_sL_s \times N_sN_f$ matrix composed by the $\boldsymbol{h}$ vector. $\boldsymbol{c}_1$, $\boldsymbol{c}_2$, ..., $\boldsymbol{c}_{N_s}$ are the code vectors of each symbol. Nevertheless, as we said, we have the same code vector for each symbol, so we can call every code vector as $\boldsymbol{c}$, whose entries are $c_1, c_2, \ldots, c_{N_f}$. In this case the matrix C in figure 3.2 contains replicas of the code matrix $\boldsymbol{C}$, whose structure is shown in figure 3.3

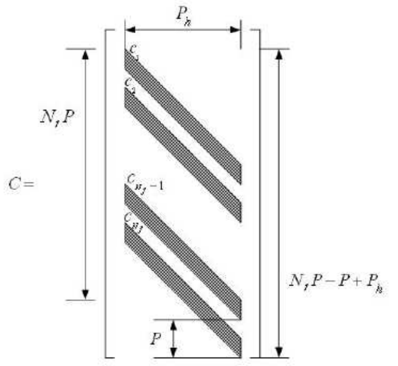

Figure 3.3: Code matrix $\boldsymbol{C}$

3.1.1 Asynchronous single user model

Without loss of generality, the unknown timing offset τ_a is in one symbol range,

$$\tau_a \in [0, N_f * T_f + T_h - T_f) \tag{3.3}$$

We can also write $\tau_a = n_aT_{sam} + T_r$, where $n_a \in \{0, 1, \ldots, N_f * P + (T_h - T_f)/Tf * P\}$ and $0 \leq T_r < T_{sam}$ is the tracking error. As said in section 2.3, the tracking error T_r can be absorbed in the unknown channel vector. So, in terms of samples, equation (3.3) becomes:

$$\tau \in \{0, 1, \ldots, L_s - 1\} \tag{3.4}$$

Let's notice that this assumption means that, stacking the first L_s received samples into a column vector, at least one of them belongs to the transmitted symbol: if $\tau = 0$, all the L_s samples are related to one symbol. If $\tau = L_s - 1$, only the last sample is the first sample of the symbol. In fact, one symbol will spread over two adjacent symbol periods. Stacking $2L_s$ data samples into a column vector to model a single symbol, we would obtain:

$$\begin{bmatrix} x[k] \\ \vdots \\ x[k+2L_s-1] \end{bmatrix} = \boldsymbol{C}_\tau \boldsymbol{h} s_k = \begin{bmatrix} \mathbf{0}_\tau \\ \boldsymbol{C} \\ \mathbf{0}_{\boldsymbol{r}} \end{bmatrix}_{2L_s \times P_h} \boldsymbol{h} s_k \tag{3.5}$$

$\boldsymbol{C}_\tau$ is made up of three blocks. $\boldsymbol{C}$ is the known user code matrix, already shown in figure 3.3. $\mathbf{0}_\tau$ is τ rows of $\mathbf{0}$. $\mathbf{0}_{\boldsymbol{r}}$ is $L_s - \tau$ rows of $\mathbf{0}$. s_k is the k_{th} symbol. But if we stack the samples into column vectors of L_s elements each one, we need two columns to describe one symbol:

$$\begin{bmatrix} x[k] & x[k+L_s] \\ \vdots & \vdots \\ x[k+Ls-1] & x[k+2L_s-1] \end{bmatrix} = \boldsymbol{C}_{s\tau} \boldsymbol{H} \boldsymbol{S} =$$

$$= \ [\boldsymbol{C}_\tau^{''} \quad \boldsymbol{C}_\tau^{'}] \begin{bmatrix} \boldsymbol{h} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{h} \end{bmatrix} \begin{bmatrix} 0 & s_k \\ s_k & 0 \end{bmatrix} \tag{3.6}$$

where $\boldsymbol{C}_{s\tau}$ is a $L_s \times 2P_h$ matrix. It's made up of two blocks. The first P_h columns constitute the matrix $\boldsymbol{C}_\tau^{''}$ and the last P_h columns form the matrix $\boldsymbol{C}_\tau^{'}$. In the matrix $\boldsymbol{C}_\tau^{''}$ the first τ rows are the last τ rows of the matrix $\boldsymbol{C}$. The other elements are all 0's. In the matrix $\boldsymbol{C}_\tau^{'}$ the last $L_s - \tau$ rows are the first $L_s - \tau$ rows of the matrix $\boldsymbol{C}$ and the other elements are 0's. See figure 3.4.
Now, remembering that the training sequence is composed of symbols all equal, we stack the received samples in a $L_s \times N_s$ matrix $\boldsymbol{X}$, extending the model in the equation (3.6) to the multi-symbol case:

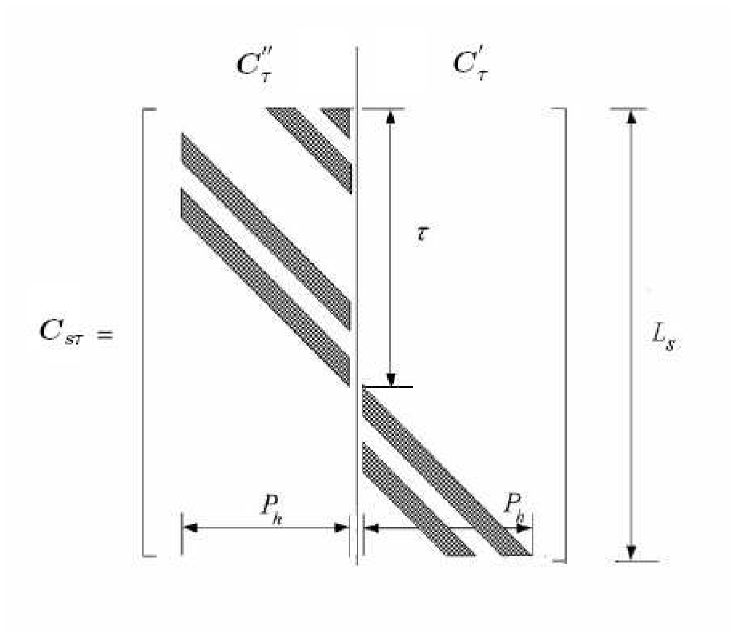

Figure 3.4: Code matrix $\boldsymbol{C}_{s\tau}$

$$
\begin{aligned}
\boldsymbol{X} &= \begin{bmatrix} x[k] & x[k+L_s] & \dots & x[k+(N_s-1)L_s] \\ \vdots & \vdots & \vdots & \vdots \\ x[k+L_s-1] & x[k+2L_s-1] & \dots & x[k+N_sL_s-1] \end{bmatrix} \\
&= \boldsymbol{C}_{s\tau}\boldsymbol{H}\boldsymbol{S} \\
&= \begin{bmatrix} \boldsymbol{C}''_\tau & \boldsymbol{C}'_\tau \end{bmatrix} \begin{bmatrix} \boldsymbol{h} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{h} \end{bmatrix} \begin{bmatrix} s_k & s_{k+1} & s_{k+2} & \dots & s_{k+N_s-1} \\ s_{k+1} & s_{k+2} & s_{k+3} & \dots & s_{k+N_s} \end{bmatrix}
\end{aligned}
$$

Let's remember that the symbols in the training sequence are equal to 1 (and notice that the training sequence is composed by $N_s + 1$ symbols, but the synchronization algorithm is applied on N_s symbol periods). In this case:

$$
\begin{aligned}
\boldsymbol{X} &= \begin{bmatrix} \boldsymbol{C}''_\tau & \boldsymbol{C}'_\tau \end{bmatrix} \begin{bmatrix} \boldsymbol{h} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{h} \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & 1 & 1 & \dots & 1 \end{bmatrix}_{2\times N_s} \\
&= \begin{bmatrix} \boldsymbol{C}''_\tau & \boldsymbol{C}'_\tau \end{bmatrix} \begin{bmatrix} \boldsymbol{h} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{h} \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} \boldsymbol{1}^T
\end{aligned}
$$

Where $\mathbf{1}^T$ is a row vector composed by N_s 1's

$$\begin{aligned}
\boldsymbol{X} &= \begin{bmatrix} \boldsymbol{C}''_\tau \boldsymbol{h} & \boldsymbol{C}'_\tau \boldsymbol{h} \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} \mathbf{1}^T \\
&= [\boldsymbol{C}_\tau]\, \boldsymbol{h}\mathbf{1}^T \qquad (3.7)
\end{aligned}$$

$\boldsymbol{C}_\tau$ is a circular shift of τ rows of the matrix $\boldsymbol{C}_0$: figure 3.5

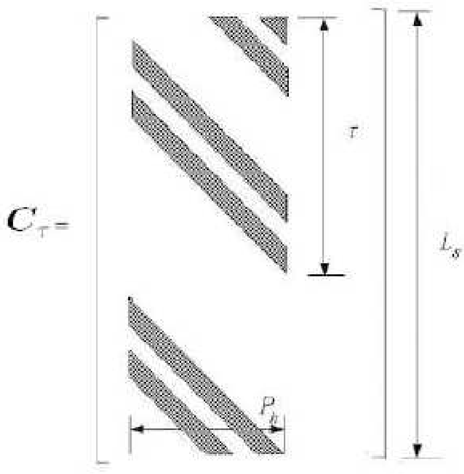

Figure 3.5: Code matrix $\boldsymbol{C}_\tau$

3.2 Synchronization algorithm

We can utilize the shift invariance property that a delay in time domain corresponds to a phase shift in frequency domain for the cyclic block $\boldsymbol{C}_\tau$: Let's define $\tilde{\boldsymbol{C}}_\tau = \mathcal{F}\boldsymbol{C}_\tau$ and $\tilde{\boldsymbol{C}}_0 = \mathcal{F}\boldsymbol{C}_0$, where the operator $\mathcal{F}$ is the FFT operation (it's a $L_s \times L_s$ FFT matrix). So, $\tilde{\boldsymbol{C}}_\tau$ is the DFT of $\boldsymbol{C}_\tau$ whereas $\tilde{\boldsymbol{C}}_0$ is the DFT of $\boldsymbol{C}_0$. Let's notice that the matrix $\boldsymbol{C}_0$ is the matrix $\boldsymbol{C}$
we can now write

$$\tilde{\boldsymbol{C}}_\tau = \tilde{\boldsymbol{C}}_0 \odot [\phi_\tau, \dots, \phi_\tau] \tag{3.8}$$

where $\phi_\tau = [1, \quad e^{-\frac{j2\pi\tau}{L_s}}, \quad \dots \quad , (e^{-\frac{j2\pi\tau}{L_s}})^{L_s-1}]^T$.
Since each column $\boldsymbol{C}_\tau$ is just one sample shift of the previous column, we can rewrite the formulation (3.8) as

$$\tilde{\boldsymbol{C}}_\tau = \tilde{\boldsymbol{c}}_{0_1}\mathbf{1}^T \odot [\phi_\tau \quad \phi_{\tau+1} \quad \dots \quad \phi_{\tau+P_h-1}] \tag{3.9}$$

where $\mathbf{1}$ is a P_h length vector with all entries equal to 1 and $\boldsymbol{c}_{0_1}$ is the first column of the code matrix $\boldsymbol{C}_0$, so that $\tilde{\boldsymbol{c}}_{0_1}$ is the first column of $\tilde{\boldsymbol{C}}_0$. Let's define $\tilde{X}_i, \quad i = 0, \dots, L_s - 1$ the elements of $\tilde{\boldsymbol{c}}_{0_1}$. They are the components of the FFT of $\boldsymbol{c}_{0_1}$:

$$\tilde{X}_i = \sum_{m=0}^{N_f-1} c_m e^{-j2\pi\frac{mPi}{L_s}} \tag{3.10}$$

because, the column $\boldsymbol{c}_{0_1}$ is:

$$\boldsymbol{c}_{0_1} = \begin{bmatrix} c_0 & 0 & 0 & c_1 & 0 & 0 & c_2 & 0 & \dots & c_{N_f-1} & 0 & \dots & 0 & 0 \end{bmatrix}_{L_s}^T$$

where, between two adjacent chips there are $(P-1)$ 0's. Let $\tilde{\boldsymbol{C}}_{inv} = [diag(\tilde{\boldsymbol{c}}_{0_1})]^{-1}$:

$$\tilde{\boldsymbol{C}}_{inv} = \begin{bmatrix} \frac{1}{\tilde{X}_0} & 0 & 0 & \dots & 0 \\ 0 & \frac{1}{\tilde{X}_1} & 0 & 0 & \vdots \\ 0 & 0 & \frac{1}{\tilde{X}_2} & 0 & \vdots \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \frac{1}{\tilde{X}_{L_s-1}} \end{bmatrix}_{L_s \times L_s}$$

We can estimate the offset τ by the following way:

$$\tilde{\boldsymbol{C}}_{inv}\tilde{\boldsymbol{C}}_\tau = [\phi_\tau \quad \phi_{\tau+1} \quad \dots \quad \phi_{\tau+P_h-1}] \tag{3.11}$$

And then, applying the IFFT operation ($\mathcal{F}^{-1}$):

$$\begin{aligned}\mathcal{F}^{-1}\tilde{\boldsymbol{C}}_{inv}\tilde{\boldsymbol{C}}_{\tau} &= \mathcal{F}^{-1}[\phi_{\tau} \quad \phi_{\tau+1} \quad \ldots \quad \phi_{\tau+P_h-1}] \\ &= \begin{bmatrix} \mathbf{0}_{\tau} \\ \boldsymbol{I}_{P_h} \\ \mathbf{0}_{r} \end{bmatrix}_{L_s \times P_h}\end{aligned} \tag{3.12}$$

$\mathbf{0}_{\tau}$ is a $\tau \times P_h$ matrix, with null elements. $\boldsymbol{I}_{P_h}$ is a $P_h \times P_h$ identity matrix. $\mathbf{0}_{\boldsymbol{r}}$ is a $r \times P_h$ matrix with null elements. This means that τ is equal to the number of rows of the matrix $\mathbf{0}_{\tau}$. (If $\tau + P_h$ is bigger than L_s then the structure is a bit different: in the matrix resulting by the equation (3.12), the first $\tau + P_h - L_s$ rows are the last $\tau + P_h - L_s$ rows of $\boldsymbol{I}_{P_h}$ and the last $L_s - \tau$ rows are the first $L_s - \tau$ rows of $\boldsymbol{I}_{P_h}$. In the middle there is the matrix $\mathbf{0}_{\tau}$). We can understand the result of equation (3.12) rewriting it as:

$$\begin{aligned}&\mathcal{F}^{-1}\tilde{\boldsymbol{C}}_{inv}\tilde{\boldsymbol{C}}_{\tau} = \\ &= \mathcal{F}^{-1}\begin{bmatrix} \frac{1}{\tilde{X}_0} & 0 & 0 & \ldots & 0 \\ 0 & \frac{1}{\tilde{X}_1} & 0 & 0 & \vdots \\ 0 & 0 & \frac{1}{\tilde{X}_2} & 0 & \vdots \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \frac{1}{\tilde{X}_{L_s-1}} \end{bmatrix} \begin{bmatrix} \tilde{X}_0 \\ \tilde{X}_1 \\ \tilde{X}_2 \\ \vdots \\ \tilde{X}_{L_s-1} \end{bmatrix} \odot [\phi_{\tau} \quad \phi_{\tau+1} \quad \ldots \quad \phi_{\tau+P_h-1}] = \\ &= \mathcal{F}^{-1}\begin{bmatrix} 1 \\ 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} \odot [\phi_{\tau} \quad \phi_{\tau+1} \quad \ldots \quad \phi_{\tau+P_h-1}] = \\ &= \mathcal{F}^{-1}[\phi_{\tau} \quad \phi_{\tau+1} \quad \ldots \quad \phi_{\tau+P_h-1}] = \\ &= \begin{bmatrix} \mathbf{0}_{\tau} \\ \boldsymbol{I}_{P_h} \\ \mathbf{0}_{r} \end{bmatrix}_{L_s \times P_h}\end{aligned}$$

Let's now apply the operation described till now to the data matrix, built with the collected samples, as in the equation (3.7):

$$\begin{aligned}
\mathcal{F}^{-1}\tilde{\boldsymbol{C}}_{inv}\mathcal{F}\boldsymbol{X} &= \mathcal{F}^{-1}\tilde{\boldsymbol{C}}_{inv}\mathcal{F}\boldsymbol{C}_{\tau}\boldsymbol{h}\boldsymbol{1}^T && (3.13)\\
&= \mathcal{F}^{-1}\tilde{\boldsymbol{C}}_{inv}\tilde{\boldsymbol{C}}_{\tau}\boldsymbol{h}\boldsymbol{1}^T\\
&= \begin{bmatrix} \boldsymbol{0}_{\tau} \\ \boldsymbol{I}_{P_h} \\ \boldsymbol{0}_{r} \end{bmatrix} \boldsymbol{h}\boldsymbol{1}^T\\
&= \begin{bmatrix} \boldsymbol{0}_{\tau} \\ \boldsymbol{h} \\ \boldsymbol{0}_{r} \end{bmatrix} \boldsymbol{1}^T && (3.14)
\end{aligned}$$

(Again, if $\tau + P_h > L_s$ the matrix in equation (3.14) is a bit different:

$$\begin{bmatrix} \boldsymbol{h}^{''} \\ \boldsymbol{0}_{\tau} \\ \boldsymbol{h}^{'} \end{bmatrix} \boldsymbol{1}^T$$

where $\boldsymbol{h}^{''}$ is the last $P_h - (L_s - \tau)$ elements of vector $\boldsymbol{h}$ and $\boldsymbol{h}^{'}$ is the first $L_s - \tau$ elements of $\boldsymbol{h}$).
Define the vector $\boldsymbol{y} = \mathcal{F}^{-1}\tilde{\boldsymbol{C}}_{inv}\mathcal{F}\boldsymbol{X}\boldsymbol{1}$:

$$\boldsymbol{y} = \begin{bmatrix} \boldsymbol{0}_{\tau} \\ N_s\boldsymbol{h} \\ \boldsymbol{0}_{r} \end{bmatrix} \tag{3.15}$$

we get the estimation of the offset:

$$\hat{\tau} = argmax_{\tau}(\frac{1}{N_s}\sum_{n=1}^{L_w}\boldsymbol{y}[\tau+n]) \quad ,\tau = 0, 1, \ldots, L_s - 1 \tag{3.16}$$

It means that, starting from the first row of $\boldsymbol{y}$ we set a window of length L_w and we sum all the elements in that window.The selection of L_w depends on the channel energy profile and the SNR: it should be a length which can include the part with the highest channel energy to noise ratio.

Chapter 4

The modified algorithm

The algorithm described in the chapter 3 was developed by Yiyin Wang, Geert Leus and Alle-Jan van der Veen. Some aspects are being improved. In this chapter we describe one problem in the algorithm and a simple practical way to solve it, modifying slightly the matrix $\tilde{\boldsymbol{C}}_{inv}$.

4.1 Noise analysis

Let's start analyzing the noise. In the equation (3.13) we didn't consider the noise. Now, let's suppose that at the antenna RX we receive the signal convolved with the channel plus White Gaussian Noise. After the first part of the receiver, the analog part, we go to integrate and dump obtaining signal plus noise samples. As we have already said, we stack the received data in the matrix $\boldsymbol{X}$, but now we define also the noise samples matrix $\boldsymbol{N}$:

$$\begin{bmatrix} n[k] & n[k+L_s] & \dots & n[k+(N_s-1)L_s] \\ \vdots & \vdots & \vdots & \vdots \\ n[k+L_s-1] & n[k+2L_s-1] & \dots & n[k+N_sL_s-1] \end{bmatrix}_{L_s \times N_s}$$

These samples are Independent Gaussian Variables, with the same mean value 0 and the same variance σ^2. This assumption has been demonstrated in [13] by Hoctor and Tomlinson. The noise matrix will undergo the same operations undergone by the matrix $\boldsymbol{X}$. First of all the FFT operation, obtaining the matrix $\tilde{\boldsymbol{N}}$:

$$\tilde{\boldsymbol{N}} = \begin{bmatrix} \tilde{n}_k[0] & \tilde{n}_{k+L_s}[0] & \dots & \tilde{n}_{k+(N_s-1)L_s}[0] \\ \vdots & \vdots & \vdots & \vdots \\ \tilde{n}_k[L_s-1] & \tilde{n}_{k+L_s}[L_s-1] & \dots & \tilde{n}_{k+(N_s-1)L_s}[L_s-1] \end{bmatrix}_{L_s \times N_s}$$

where

$$\tilde{n}_k[i] = \sum_{n=0}^{L_s-1} n_{k+n} e^{-j2\pi\frac{ni}{L_s}} \qquad , i = 0, 1, \dots, 2L_s - 1 \tag{4.1}$$

The sum of indipendent Gaussian Variables is still a Gaussian Variable. Its mean value is the sum of all mean values, so it's 0. Let's calculate the variance:

$$\begin{aligned} \sigma^2_{\tilde{n}_k[i]} &= E\{(\tilde{n}_k[i]) \cdot (\tilde{n}_k[i])^*\} \\ &= E\{(\sum_{n=0}^{L_s-1} n_{k+n} e^{-j2\pi\frac{ni}{L_s}}) \cdot (\sum_{l=0}^{L_s-1} n_{k+l} e^{-j2\pi\frac{li}{L_s}})^*\} \\ &= E\{\sum_{n=0}^{L_s-1}\sum_{l=0}^{L_s-1} n_{k+n} n^*_{k+l} e^{-j2\pi\frac{ni}{L_s}} e^{j2\pi\frac{li}{L_s}}\} \\ &= \sum_{n=0}^{L_s-1}\sum_{l=0}^{L_s-1} E\{n_{k+n} n^*_{k+l}\} e^{-j2\pi\frac{ni}{L_s}} e^{j2\pi\frac{li}{L_s}} \end{aligned}$$

We know that the variables are independent, so $E\{n_{k+n}n^*_{k+l}\} \neq 0$ only for $l = n$:

$$\sigma^2_{\tilde{n}_k[i]} = \sum_{n=0}^{L_s-1} E\{|n_{k+n}|^2\} = \sum_{n=0}^{L_s-1} \sigma^2 = L_s\sigma^2 \tag{4.2}$$

So, each element of $\tilde{\boldsymbol{N}}$ is a Gaussian variable with mean value 0 and variance $L_s\sigma^2$. The second step is the multiplication with $\tilde{\boldsymbol{C}}_{inv}$:

$$\tilde{\boldsymbol{N}}' = \tilde{\boldsymbol{C}}_{inv}\tilde{\boldsymbol{N}} = \tag{4.3}$$

$$= \begin{bmatrix} \frac{1}{\tilde{X}_0} & 0 & 0 & \dots & 0 \\ 0 & \frac{1}{\tilde{X}_1} & 0 & 0 & \vdots \\ 0 & 0 & \frac{1}{\tilde{X}_2} & 0 & \vdots \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \frac{1}{\tilde{X}_{L_s-1}} \end{bmatrix} \begin{bmatrix} \tilde{n}_k[0] & \tilde{n}_{k+L_s}[0] & \dots & \tilde{n}_{k+(N_s-1)L_s}[0] \\ \vdots & \vdots & \vdots & \vdots \\ \tilde{n}_k[L_s-1] & \tilde{n}_{k+L_s}[L_s-1] & \dots & \tilde{n}_{k+(N_s-1)L_s}[L_s-1] \end{bmatrix}$$

$$= \begin{bmatrix} \frac{\tilde{n}_k[0]}{\tilde{X}_0} & \frac{\tilde{n}_{k+L_s}[0]}{\tilde{X}_0} & \dots & \frac{\tilde{n}_{k+(N_s-1)L_s}[0]}{\tilde{X}_0} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\tilde{n}_k[L_s-1]}{\tilde{X}_{L_s-1}} & \frac{\tilde{n}_{k+L_s}[L_s-1]}{\tilde{X}_{L_s-1}} & \dots & \frac{\tilde{n}_{k+(N_s-1)L_s}[L_s-1]}{\tilde{X}_{L_s-1}} \end{bmatrix}_{L_s \times N_s}$$

And finally we apply the IFFT to this matrix, obtaining the matrix that we call $\boldsymbol{N}'$:

$$\boldsymbol{N}' = \mathcal{F}^{-1}\tilde{\boldsymbol{N}}' = \mathcal{F}^{-1}\tilde{\boldsymbol{C}}_{inv}\tilde{\boldsymbol{N}}$$

$$= \begin{bmatrix} n'_k & n'_{k+L_s} & \dots & n'_{k+(N_s-1)L_s} \\ n'_{k+1} & n'_{k+L_s+1} & \dots & \vdots \\ \vdots & \vdots & \vdots & \vdots \\ n'_{k+L_s-1} & n'_{k+2L_s-1} & \dots & n'_{k+N_sL_s-1} \end{bmatrix}_{L_s \times N_s}$$

where, for the first column entries we have:

$$n'_{k+m} = \frac{1}{L_s}\sum_{i=0}^{L_s-1} \tilde{n}_k[i]e^{j2\pi\frac{mi}{L_s}} \qquad , m = 0, 1, \dots, L_s - 1$$

It's the same for the other columns: for example for the second column we have to write:

$$n'_{k+L_s+m} = \frac{1}{L_s}\sum_{i=0}^{L_s-1} \tilde{n}_{k+L_s}[i]e^{j2\pi\frac{mi}{L_s}} \qquad , m = 0, 1, \dots, L_s - 1$$

Now, imagine to apply the IFFT to the matrix $\tilde{\boldsymbol{N}}$. We would obtain again the matrix $\boldsymbol{N}$, which has independent Gaussian variables as elements. In fact, we apply the IFFT to the matrix $\tilde{\boldsymbol{N}}' = \tilde{\boldsymbol{C}}_{inv}\tilde{\boldsymbol{N}}$, but the matrix $\tilde{\boldsymbol{C}}_{inv}$ is a deterministic quantity: it means that the elements in the matrix $\boldsymbol{N}'$ are still independent Gaussian variables. They still have the same mean value equal to 0. But now the variance of each element is different from the others. So, let's calculate the variance of each element of the matrix $\boldsymbol{N}'$:

$$\begin{aligned}\sigma^2_{n'_{k+m}} &= E[(n'_{k+m}) \cdot (n'_{k+m})^*] \\ &= E[\frac{1}{(L_s)^2}(\sum_{i=0}^{L_s-1} \frac{\tilde{n}_k[i]}{\tilde{X}_i} e^{j2\pi\frac{im}{L_s}})(\sum_{t=0}^{L_s-1} \frac{\tilde{n}_k^*[t]}{\tilde{X}_t^*} e^{-j2\pi\frac{tm}{L_s}})]\end{aligned} \tag{4.4}$$

$$\begin{aligned}&= E[\frac{1}{(L_s)^2}(\sum_{i=0}^{L_s-1} \frac{\sum_{n=0}^{L_s-1} n_{k+n} e^{-j2\pi\frac{ni}{L_s}}}{\tilde{X}_i} e^{j2\pi\frac{im}{L_s}}) \cdot \\ &\qquad \cdot(\sum_{t=0}^{L_s-1} \frac{\sum_{l=0}^{L_s-1} n^*_{k+l} e^{+j2\pi\frac{lt}{L_s}}}{\tilde{X}_t^*} e^{-j2\pi\frac{tm}{L_s}})] \\ &= \frac{1}{(L_s)^2}\sum_{i=0}^{L_s-1}\sum_{t=0}^{L_s-1} \frac{\sum_{n=0}^{L_s-1}\sum_{l=0}^{L_s-1} E[n_{k+n}n^*_{k+l}] e^{-j2\pi\frac{ni-lt}{L_s}}}{\tilde{X}_i\tilde{X}_t^*} e^{j2\pi\frac{m(i-t)}{L_s}} \\ &= \frac{1}{(L_s)^2}\sum_{i=0}^{L_s-1}\sum_{t=0}^{L_s-1} \frac{\sum_{n=0}^{L_s-1} E[|n_{k+n}|^2] e^{-j2\pi\frac{n(i-t)}{L_s}}}{\tilde{X}_i\tilde{X}_t^*} e^{j2\pi\frac{m(i-t)}{L_s}} \\ &= \frac{1}{(L_s)^2}\sum_{i=0}^{L_s-1}\sum_{t=0}^{L_s-1} \frac{\sum_{n=0}^{L_s-1} \sigma^2 e^{-j2\pi\frac{n(i-t)}{L_s}}}{\tilde{X}_i\tilde{X}_t^*} e^{j2\pi\frac{m(i-t)}{L_s}} \\ &= \frac{\sigma^2}{(L_s)^2}\sum_{n=0}^{L_s-1}(\sum_{i=0}^{L_s-1}\frac{1}{\tilde{X}_i}e^{+j2\pi\frac{(m-n)i}{L_s}} \sum_{t=0}^{L_s-1}\frac{1}{\tilde{X}_t^*}e^{-j2\pi\frac{(m-n)t}{L_s}}) \\ &= \frac{\sigma^2}{(L_s)^2}\sum_{n=0}^{L_s-1}[(\sum_{i=0}^{L_s-1}\frac{1}{\tilde{X}_i}e^{+j2\pi\frac{(m-n)i}{L_s}})(\sum_{t=0}^{L_s-1}\frac{1}{\tilde{X}_t}e^{+j2\pi\frac{(m-n)t}{L_s}})^*] \\ &= \frac{\sigma^2}{(L_s)^2}\sum_{n=0}^{L_s-1}|\sum_{i=0}^{L_s-1}\frac{1}{\tilde{X}_i}e^{+j2\pi\frac{(m-n)i}{L_s}}|^2 \quad \leq \quad \frac{\sigma^2}{L_s}\sum_{i=0}^{L_s-1}\frac{1}{|\tilde{X}_i|^2}\end{aligned} \tag{4.5}$$

As we can see, the variance changes elements by elements and also the value is different for different code sequences. We can only say that in the matrix $\boldsymbol{N}'$ the m_{th} element of each column has the same variance as the m_{th} element of each other column. Looking at equation (4.5), we realize that the most important result is that now we know that the variance could increase as the inverse of the norm of the components $\tilde{X}_i$ becomes larger. We would like to have components $\tilde{X}_i$ with norm high enough to avoid that the variance of the noise becomes too high.

4.2 Error probability and MSE

In order to understand the advantage to have components $\tilde{X}_i$ with high norm, we must analyze how the variance of the noise influences the estimation of the offset τ. Remembering the formula (2.16), now we introduce also the noise. So, the vector $\boldsymbol{y}$ now contains also noise. In fact, define:

$$\boldsymbol{Z} = \boldsymbol{X} + \boldsymbol{N}$$

we apply all the operations described before to the matrix $\boldsymbol{Z}$:

$$\begin{aligned}
\mathcal{F}^{-1}\tilde{\boldsymbol{C}}_{inv}\mathcal{F}\boldsymbol{Z} &= \mathcal{F}^{-1}\tilde{\boldsymbol{C}}_{inv}\mathcal{F}(\boldsymbol{X}+\boldsymbol{N}) \\
&= \mathcal{F}^{-1}\tilde{\boldsymbol{C}}_{inv}\mathcal{F}\boldsymbol{X} + \mathcal{F}^{-1}\tilde{\boldsymbol{C}}_{inv}\mathcal{F}\boldsymbol{N} \\
&= \begin{bmatrix} \boldsymbol{0}_\tau \\ \boldsymbol{I}_{P_h} \\ \boldsymbol{0}_r \end{bmatrix} \boldsymbol{h}\boldsymbol{1}^T + \boldsymbol{N}'
\end{aligned}$$

Let's define

$$v(\tau) = \frac{1}{N_s}\sum_{n=1}^{T_w} \boldsymbol{y}[\tau+n] \quad , \tau = 0, \ldots, L_s - 1$$

and

$$\boldsymbol{v} = [v(0) \quad v(1) \quad \dots \quad v(L_s - 1)]$$

remembering the formula (3.16) and supposing that the offset is equal to τ_0, we have the correct estimation when $v(\tau_0)$ is the maximum element of the vector $\boldsymbol{v}$. We must calculate the statistics of $v(\tau_0)$.

$$v(\tau_0) = \sum_{n=1}^{L_w} \left(h[n-1] + \frac{1}{N_s} \sum_{i=0}^{N_s - 1} n'_{k+\tau_0+(n-1)+iL_s} \right) \tag{4.6}$$

$v(\tau_0)$ is the sum of a deterministic variable $\sum_{n=1}^{L_w} h[n-1]$ plus the sum of $L_s N_s$ independent gaussian variables with zero mean value and different variances. So, $v(\tau_0)$ is still a gaussian variable with mean value equal to $\sum_{n=1}^{L_w} h[n-1]$ and variance equal to the sum of the variances of the $L_s N_s$ noise elements, divided by N_s^2. When τ differs from τ_0, the reasoning is exactly the same, but the mean value is smaller and the variance is different, because, as shown in section 2.1, the elements of matrix $\boldsymbol{N}'$ have in general different variances. We can write:

$$v(\tau_0) \in \mathcal{N}(\mu_{\tau_0}, \sigma^2_{\tau_0})$$

$$v(\tau) \in \mathcal{N}(\mu_{\tau}, \sigma^2_{\tau}) \quad , \tau = 0, 1, \dots, L_s - 1 \quad \tau \neq \tau_0$$

where we know only that μ_{τ_0} is bigger than μ_τ, for $\tau \neq \tau_0$. The estimation of the offset is correct when

$$v(\tau_0) > v(\tau), \quad \tau = 0, 1, \dots, L_s - 1 \quad \tau \neq \tau_0 \tag{4.7}$$

So, the probability of correct estimation is equal to $(1 - Pr\{error\})$:

$$\begin{aligned}
1 - Pr\{error\} &= 1 - (Pr\{v(0) > v(\tau_0)\} + \ldots + Pr\{v(L_s - 1) > v(\tau_0)\}) \\
&= 1 - (Pr\{v(0) - v(\tau_0) > 0\} + \ldots + Pr\{v(L_s - 1) - v(\tau_0) > 0\}) \\
&= 1 - (Q\left(\frac{\mu_{\tau_0} - \mu_0}{\frac{1}{N_s}\sqrt{\sigma_{\tau_0}^2 + \sigma_0^2}}\right) + \ldots + Q\left(\frac{\mu_{\tau_0} - \mu_{L_s-1}}{\frac{1}{N_s}\sqrt{\sigma_{\tau_0}^2 + \sigma_{L_s-1}^2}}\right)) \\
&= 1 - \sum_{\substack{\tau=0 \\ \tau\neq\tau_0}}^{L_s-1} Q\left(\frac{\mu_{\tau_0} - \mu_\tau}{\frac{1}{N_s}\sqrt{\sigma_{\tau_0}^2 + \sigma_\tau^2}}\right) \qquad (4.8)
\end{aligned}$$

The result of equation (4.8) makes us understand the importance to have small variances. We can also calculate the normalized Mean Squared Error of the estimation of the offset as:

$$MSE = E\left[\left(\frac{\tilde{\tau} - \tau_0}{L_s}\right)^2\right] = \sum_{\tau=0}^{L_s-1} Pr(\tilde{\tau} = \tau)\left(\frac{\tau - \tau_0}{L_s}\right)^2 \qquad (4.9)$$

To calculate the probabilities $Pr(\tilde{\tau} = \tau)$ of each term in the summation in equation (4.9) we have just to notice that they have similar expressions as the probability of correct estimation (4.8). We arrive to the same conclusion: we want to have small variances of the noise elements in the matrix $\boldsymbol{N}'$. If we generate the code sequence randomly, as a random sequence of +1 and -1, we don't have any control on the consequences. This means that we could have also components $\tilde{X}_i$ with very small norm, also equal to 0 in some cases. In the next section we'll describe a practical method to avoid this problem, also keeping the code sequences randomly generated.

4.3 The solution: replacement of components

The first approach to solve the problem described in the preceding section was to work on the design of the code sequences. The idea was to propose a particular configuration for the code sequences, maybe also redefining the set of values to which the chips belong, for example, a particular set of complex numbers with certain phases and certain norms. The goal of the new design was to avoid to produce DFT vectors with components with too small norms. The problem met during this approach is the following: the FFT depends on many parameters and even if we find a sequence that produces a DFT with components characterized by norms high enough, the same sequence could produce a completely different result when the parameters change. In fact, remembering the equation (3.10), we notice that $\tilde{X}_i$ depends on the parameters P, P_h, N_f ($L_s = (N_f - 1)P + P_h$). Changing one of them, we change the result of the FFT. Actually there exist some code sequence that produce DFT with constant norms. But in our algorithm we apply the FFT operation to a vector that is a bit different by the transmitted code sequence. In fact, let's suppose that the vector

$$\boldsymbol{c} = \begin{bmatrix} c_0 & c_1 & c_2 & \dots & c_{N_f-1} \end{bmatrix}$$

is a sequence that has a DFT with constant norms. In our algorithm the FFT operation is applied to the first column of the matrix $\boldsymbol{C}_0$. This column has the following structure:

$$\boldsymbol{c}_{0_1} = \begin{bmatrix} c_0 & 0 & 0 & c_1 & 0 & 0 & c_2 & 0 & \dots & c_{N_f-1} & 0 & \dots & 0 & 0 \end{bmatrix}^T_{L_s}$$

It's not just an oversampling of the vector $\boldsymbol{c}$, because between two chips there are $(P - 1)$ 0's, but after the last chip c_{N_f-1} there are $P_h - 1$ 0's. It would be an oversampling if $P_h = P$. But this means that we wouldn't have Inter Frame Interference, because $P_h = P$ means $T_h = T_f$ whereas we made the assumptions that we have IFI and T_h could be also $\gg T_f$. If $\boldsymbol{c}_{0_1}$ was just an oversampling of $\boldsymbol{c}$ we would have a DFT with still with constant norms. But in reality, we don't have just an oversampling and in fact the DFT changes completely.
The solution was found following another approach. The idea was simply to replace those components $\tilde{X}_i$ that are too small, compared with the other components. In fact, the ideal situation is when all the frequency components have more or less the same energy, around a certain mean value. But there are situations in which some components have too small energy and this means that the variance could become too high, as we can understand looking at (4.5). So, the idea is to calculate the FFT of the column vector $\boldsymbol{c}_{0_1}$. Then calculate the inverse of each components $\tilde{X}_i$. After that, set a threshold under a certain criteria and finally replace the elements $\frac{1}{\tilde{X}_i}$, that are over the threshold, with another element (for example 0). In this way we'll reduce the value of the variances of the elements in $\boldsymbol{N}'$, but we'll generate also a mismatch. In fact, let's take a look to the result we have modifing the algorithm

as described: instead of the matrix $\tilde{\boldsymbol{C}}_{inv}$ of equation (3.12), now we have the matrix $\tilde{\boldsymbol{C}}_{inv\_new}$

$$\tilde{\boldsymbol{C}}_{inv\_new} = \begin{bmatrix} \frac{1}{\tilde{X}_0} & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 0 & \dots & \vdots \\ 0 & 0 & \frac{1}{\tilde{X}_2} & 0 & \dots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & \frac{1}{\tilde{X}_{L_s-2}} & 0 \\ 0 & 0 & 0 & \dots & 0 & 0 \end{bmatrix}$$

In this example we replaced the elements $\frac{1}{\tilde{X}_1}$ and $\frac{1}{\tilde{X}_{L_s-2}}$ with 0's. Let's substitute in equation (3.12) the matrix $\tilde{\boldsymbol{C}}_{inv}$ with $\tilde{\boldsymbol{C}}_{inv\_new}$:

$$\begin{aligned} &\tilde{\boldsymbol{C}}_{inv\_new}\tilde{C}_{\mathcal{T}} = \qquad (4.10) \\ &= \begin{bmatrix} \frac{1}{\tilde{X}_0} & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & \vdots \\ 0 & 0 & \frac{1}{\tilde{X}_2} & \dots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 0 \end{bmatrix} \begin{bmatrix} \tilde{X}_0 \\ \tilde{X}_1 \\ \tilde{X}_2 \\ \vdots \\ \tilde{X}_{L_s-1} \end{bmatrix} 1^T \odot [\phi_{\mathcal{T}} \quad \phi_{\mathcal{T}+1} \quad \dots \quad \phi_{\mathcal{T}+P_h-1}] \\ &= \begin{bmatrix} 1 \\ 0 \\ 1 \\ \vdots \\ 1 \\ 0 \end{bmatrix} 1^T \odot [\phi_{\mathcal{T}} \quad \phi_{\mathcal{T}+1} \quad \dots \quad \phi_{\mathcal{T}+P_h-1}] \\ &= \left\{ \begin{bmatrix} 1 \\ 1 \\ 1 \\ \vdots \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} \right\} 1^T \odot [\phi_{\mathcal{T}} \quad \phi_{\mathcal{T}+1} \quad \dots \quad \phi_{\mathcal{T}+P_h-1}] \end{aligned}$$

$$= \begin{bmatrix} 1 \\ 1 \\ 1 \\ \vdots \\ 1 \\ 1 \end{bmatrix} 1^T \odot [\phi_\tau \quad \phi_{\tau+1} \quad \dots \quad \phi_{\tau+P_h-1}] - \begin{bmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} 1^T \odot [\phi_\tau \quad \phi_{\tau+1} \quad \dots \quad \phi_{\tau+P_h-1}]$$

Then, applying the IFFT, we obtain the same result as before, plus a new matrix that we call $\boldsymbol{M}$:

$$\tilde{\boldsymbol{C}}_{inv\_new}\tilde{C}_\tau = \begin{bmatrix} \boldsymbol{0}_\tau \\ \boldsymbol{I}_{P_h} \\ \boldsymbol{0_r} \end{bmatrix} + \boldsymbol{M} \tag{4.11}$$

where:

$$\begin{aligned} \boldsymbol{M} &= \mathcal{F}^{-1}\left\{ - \begin{bmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} \boldsymbol{1}^T \odot [\phi_\tau \quad \phi_{\tau+1} \quad \dots \quad \phi_{\tau+P_h-1}] \right\} \qquad (4.12) \\ &= \mathcal{F}^{-1}\left\{ -\boldsymbol{\varepsilon} \cdot 1^T \odot [\phi_\tau \quad \phi_{\tau+1} \quad \dots \quad \phi_{\tau+P_h-1}] \right\} \end{aligned}$$

So, let's replace the *new* result (4.11) to the *old* result (3.12):

$$
\begin{aligned}
\mathcal{F}^{-1}\{\tilde{\boldsymbol{C}}_{inv\_new}\mathcal{F}\boldsymbol{X}\} &= \\
&= \left(\begin{bmatrix} \boldsymbol{0}_{\tau} \\ \boldsymbol{I}_{P_h} \\ \boldsymbol{0}_{\boldsymbol{r}} \end{bmatrix} + \boldsymbol{M} \right) \boldsymbol{h}\boldsymbol{1}^T \\
&= \begin{bmatrix} \boldsymbol{0}_{\tau} \\ \boldsymbol{h} \\ \boldsymbol{0}_{\boldsymbol{r}} \end{bmatrix} \boldsymbol{1}^T + \Delta \qquad (4.13)
\end{aligned}
$$

where

$$\Delta = \boldsymbol{M}\boldsymbol{h}\boldsymbol{1}^T \qquad (4.14)$$

The matrix Δ depends on the number of components that we replace and also the way to replace these components. In fact replacing the component $\frac{1}{\tilde{X}_i}$ with 0 is not the best choice. To set the threshold and also to define the new elements to put in the matrix $\tilde{\boldsymbol{C}}_{inv\_new}$, we should follow some criteria, as zero forcing or a criteria to minimize the MSE, or something else. In this thesis a practical method to generate the matrix $\tilde{\boldsymbol{C}}_{inv\_new}$ is proposed, looking at values that we obtain by simulations. Observing the norms of the components $\frac{1}{\tilde{X}_i}$ we have noted that in every case, changing sequences and parameters, we don't have many components that become too high, compared to the total number of components. It means that the vector $\boldsymbol{\varepsilon}$ in (4.12) will have few elements different than 0, compared to the total number of elements. But now we should understand what does *high* components mean. We need a threshold and then define *too high* the components $\frac{1}{\tilde{X}_i}$ that have a norm bigger than the threshold. The proposed method is to treat the norms of the components $\frac{1}{\tilde{X}_i}$ as random variables with a certain average and a certain standard deviation. We have good sequences when the energy of the frequency components of $\boldsymbol{c}_{0_1}$ is more or less the same, around a certain mean value. Otherwise, a bad sequence produces a vector $\boldsymbol{c}_{0_1}$ that has some frequency components too small and this means to have some peaks in the vector $\boldsymbol{u}$ defined as

$$\boldsymbol{u} = \left\{ \frac{1}{|\tilde{X}_i|} \right\}_{i=0}^{L_s-1}$$

We have also to remember that the same sequence could be bad or good, changing some parameters P, P_h, N_f.
So, for each sequence $\boldsymbol{u}$, we calculate the average and the standard deviation of the elements. Then we set the threshold: looking at the results of the simulations, we found the best threshold was equal to the average plus standard deviation. Finally we replace those components having norm bigger than the threshold with new components that have the same phases but norms equal to the average. This choice is due to the fact that we want to have a matrix Δ with negligible values and replacing the components with zero is a poor choice, besides besides that it is not necessary. The last thing that we have to understand is that the peaks that really can increase the variance of the noise samples are present in those vectors $\boldsymbol{u}$ in which more or less the values are all around the mean value, except on these few peaks. Otherwise, we can also have many peaks but not so high to worsen the performance really. The following figure helps to understand:

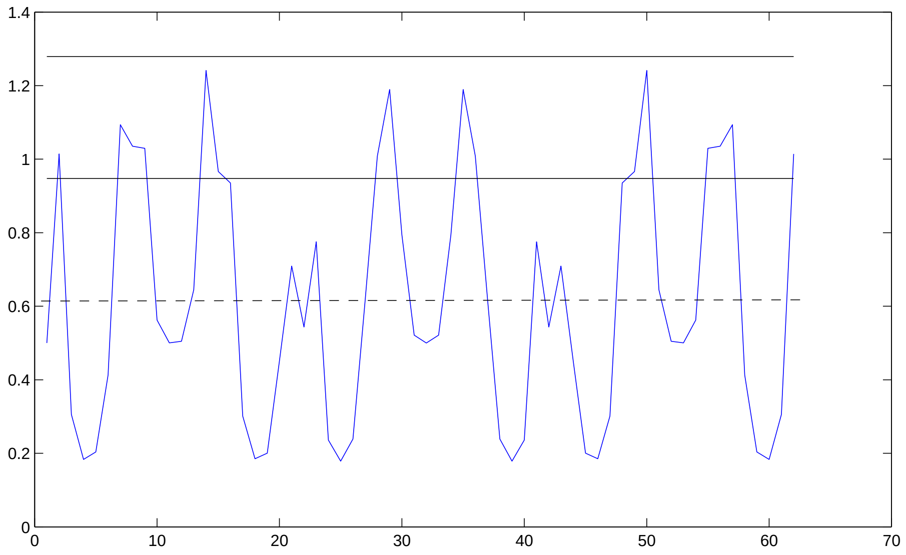

Figure 4.1: Vector $\boldsymbol{u}$

In figure 4.1 we can see plotted the vector $\boldsymbol{u}$: the related code sequence is randomly generated, the parameters are $P = 3, T_f = 30ns, T_h = 100ns, N_f = 8$. The dashed line is the mean value. The under solid line is the Threshold equal to the average + standard deviation. The upper solid line is the average + 2 standard deviation. As we can see, many elements of the vector are above the threshold. Nevertheless this sequence is a good sequence, the values they have aren't really so high: we have problems when we have only few peaks with high values. For example let's look at figure 4.2. In this figure now we can see much less peaks over the threshold, but

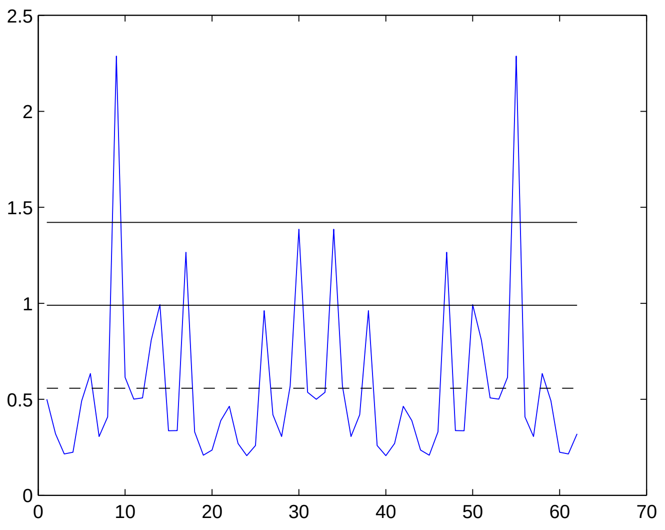

Figure 4.2: Vector $\boldsymbol{u}$

with higher values: we have also to remember that these values will squared.
So, in practical cases the last check to make is to see if there are values also over the upper solid line (average + 2 standard deviation). If no elements are over the upper solid line we consider the sequence a good sequence even if some components are over the threshold. Otherwise, when we find components over the upper solid line, we consider the sequence a bad sequence and we replace all the components over the threshold. The explained method is just a practical way found by simulations. In the next section we can see the gain reached with this method.

4.4 Simulation results

The simulations are made with the following parameters:
$T_f = 30ns, T_h = 90ns, N_f = 15, P = 3, N_s = 30$. The code sequences are randomly generated. We made 1000 Monte Carlo runs: In each run, the timing offset and the

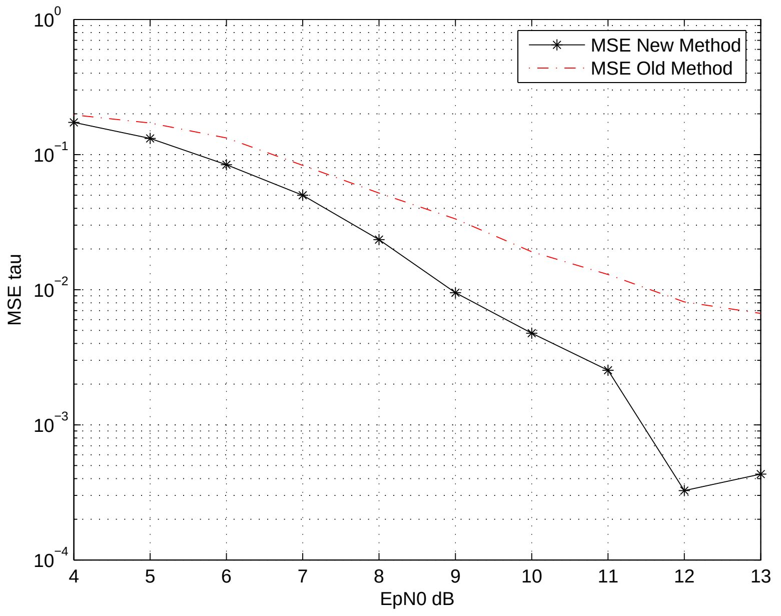

Figure 4.3: MSE new method compared with the old one

channel are randomly generated. In figure we can see the MSE of the estimation of the offset (defined in (4.9)) for some values of ratio $\frac{E_p}{N_0}$. The dashed curve is related to the performance for the original algorithm. The solid curve is plotted with the same sequences but with the modification explained in this chapter. Other simulations were done, with different parameters, but similar results. The method could be improved following some more theorical criteria.

Chapter 5

Detection Theory

Detection theory is defined as how to make a decision if an event of interest is present or not [14]. In detection theory for TR-UWB Systems we want to determine whether data is present or not. This results in a binary hypothesis test, where two cases (hypotheses) are stated and the algorithm has to decide which one is (most likely) true.
In the synchronization algorithm described in chapter 3 the signal is supposed to be present starting from the first column of the data matrix built by collecting the received samples. But if the signal is not present, the data model (3.7) is not valid anymore and the synchronization algorithm doesn't produce a correct estimation of the offset. In this chapter we propose how to detect the presence of the signal before applying the synchronization algorithm, in the single user case.

5.1 The noise samples

The mentioned model (3.7) has to be completed, because, as said in section 4.1, we must consider the presence of the noise. The assumption we make is that the noise picked up by the antenna receiver is an Additive Gaussian White Noise (AWGN).

Looking at figure 5.1 we notice now the presence of the Band Pass filter after the antenna receiver. Usually the filter is omitted. As said, we suppose $W_r(t)$ be AWGN, with Power Spectral Density equal to N_0. The Band Pass filter keeps only the noise in the band of the signal. Let's suppose the signal frequency be f_0 and the radio frequency signal band be β: in figure 5.2 we can see the band pass filter frequency response.
So, the Power Spectral Density of the noise $W(t)$, after the BP filter, is represented

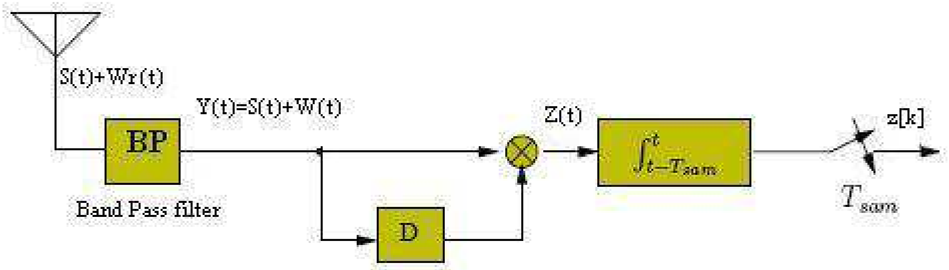

Figure 5.1: Autocorrelation receiver with Band Pass filter

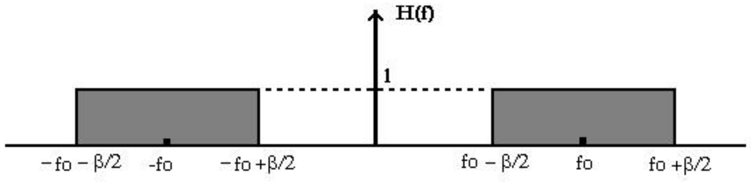

Figure 5.2: Frequency response Band Pass filter

in the figure 5.3.

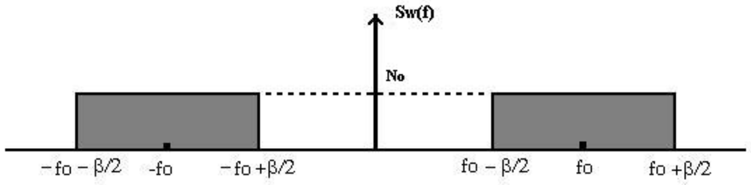

Figure 5.3: Noise power spectral density

The average power of the noise $W(t)$ is

$$N = \int_{-\infty}^{+\infty} S_w(f)df = 2N_0\beta \tag{5.1}$$

Now, we must distinguish 2 cases:

$$hypothesis \quad H_0 = \{There \quad isn't \quad the \quad signal\}$$

$$hypothesis \quad H_1 = \{The \quad signal \quad is \quad present\}$$

In the first case H_0 the expression of $Y(t)$ is:

$$Y(t) = W(t)$$

In the second case H_1:

$$Y(t) = S(t) + W(t)$$

Under hypothesis H_0 we have:

$$Z(t) = W(t)W(t-D) \tag{5.2}$$

there is only the noise, it's only one term. Under Hypothesis H_1 we have:

$$\begin{aligned} Z(t) &= [S(t)+W(t)][S(t-D)+W(t-D)] \qquad (5.3) \\ &= S(t)S(t-D) + [S(t)W(t-D) + S(t-D)W(t) + W(t)W(t-D)] \end{aligned}$$

there is signal plus noise, they are four terms, but three of them are noise. Then there's the integration and the sampling. We already know the expression of the signal samples after the oversampling. What we want now is to know the statistics of the noise samples. A good analysis is made by Hoctor and Tomlinson in [13] . From this paper we can assert that in both the hypothesis H_0 and H_1 the noise samples are INDEPENDENT GAUSSIAN variables, with null mean value and the same variance. Nevertheless the value of the variance is different in the two different hypothesis. In the hypothesis H_0 the expression of the variance is [13]

$$\sigma^2_{H_0} = 4T_{sam}N_0^2\beta \tag{5.4}$$

where T_{sam} is the integration interval and also the sampling interval. In the hypothesis H_1 the variance depends also on the power of the signal, in fact the expression of the variance is

$$\sigma^2_{H_1} = 4T_{sam}(1+\frac{S}{N})N_0^2\beta \tag{5.5}$$

where $\frac{S}{N}$ is the signal to noise power ratio at the input of the pulse pair correlator (N has the expression (5.1)).
Now we can decide a strategy of signal detection.

5.2 The training sequence for the first stage

Considering the model (3.7), when the synchronization algorithm is applied, the signal is present starting from the first column of the matrix $\boldsymbol{X}$. As already said: $\boldsymbol{X}$ is a $L_s \times N_s$ data matrix and it is equal to $[\boldsymbol{C}_\tau]\boldsymbol{h}\boldsymbol{1}^T$ when the training sequence used for the synchronization is composed of $N_s + 1$ symbols all equal to 1.
So, before applying the synchronization algorithm we have to be sure that the signal is present since the first column. To do it, a good idea is to transmit a previous training sequence composed of N_d symbols still equal to 1, but with another particular characteristic: the chips of the code sequence are all equal to 1. We can define it as the "first stage": the detection of the presence of the signal.
Transmitting the first training sequence and collecting the samples in the same way we do in the synchronization algorithm we obtain:

$$\boldsymbol{X} = [\boldsymbol{C}_\tau]\,\boldsymbol{h}\boldsymbol{1}^T \tag{5.6}$$

that is the same expression of (3.7) used to apply the synchronization algorithm, but in this case $\boldsymbol{1}^T$ is a N_d length vector and in $\boldsymbol{C}_\tau$ the elements different by zero are all equal to 1. Let's think for a moment to sum all the elements of the matrix $\boldsymbol{X}$, defined in (3.7):

$$\sum_{n=0}^{L_s N_d - 1} x[k+n] = N_f E_h N_d \tag{5.7}$$

where

$$E_h = \sum_{i=0}^{P_h - 1} h[i] \tag{5.8}$$

To understand it, let's look at the following example, in which $N_f = 3$, $P = 2$, $Ph = 6$ and $\tau = 3$:

$$\begin{aligned}
\boldsymbol{X} &= [\boldsymbol{C}_\tau]\boldsymbol{h}\mathbf{1}^T \\
&= \begin{bmatrix} 0&0&0&1&0&0\\ 0&0&0&0&1&0\\ 0&0&0&0&0&1\\ 1&0&0&0&0&0\\ 0&1&0&0&0&0\\ 0&0&1&0&0&0\\ 1&0&0&1&0&0\\ 0&1&0&0&1&0\\ 0&0&1&0&0&1\\ 1&0&0&1&0&0\\ 0&1&0&0&1&0\\ 0&0&1&0&0&1 \end{bmatrix} \begin{bmatrix} h(0)\\ h(1)\\ \vdots\\ h(5) \end{bmatrix} \mathbf{1}^T
\end{aligned}$$

$$\boldsymbol{X} = \begin{bmatrix} h(3)\\ h(4)\\ h(5)\\ h(0)\\ h(1)\\ h(2)\\ h(0)+h(3)\\ h(1)+h(4)\\ h(2)+h(5)\\ h(0)+h(3)\\ h(1)+h(4)\\ h(2)+h(5) \end{bmatrix} \mathbf{1}^T$$

If we sum the elements of each column, we obtain
$3\sum_{n=0}^{5} h(n)\mathbf{1}^T$ or in general $N_f \sum_{n=0}^{P_h-1} h(n)\mathbf{1}^T = N_f E_h \mathbf{1}^T$
So that summing all the elements we obtain the expression (5.7). We'll use this result later. Now in the next section we'll define a criteria to set a threshold to decide if the signal is present or not.

5.3 The Neyman-Pearson theorem

Let's start introducing a general method to detect the presence of a signal, as done also in [8].
Following [14] let's consider to send a signal with constant amplitude A in white Gaussian noise $w[n]$ with variance σ^2. First of all, we develop the data model and the hypothesis for this specific case. When the signal is not present we are in the "noise only" case, the so called "null hypothesis" , which we indicate with H_0, whereas H_1 is the "signal embedded in noise" case, the so called "alternative hypothesis".

$$\begin{aligned} H_0 : &\quad z[n] = w[n] \\ H_1 : &\quad z[n] = A + w[n] \end{aligned}$$

We need to define a threshold in way to decide if the received samples $z[n]$ belong to the hypothesis H_0 or H_1. To do it we need to establish a certain detection criteria. The performance of a detector can be characterized by its probability of correct detection (P_D) and false alarm rate (P_{FA}):

$$\begin{aligned} P_{FA} &= \mathit{Probability\ of\ deciding\ } H_1 \mathit{\ when\ } H_0 \mathit{\ is\ true} \\ P_D &= \mathit{Probability\ of\ deciding\ } H_0 \mathit{\ when\ } H_0 \mathit{\ is\ true} \end{aligned}$$

It's intuitive to understand that the two mentioned probabilities are correlated: when the probability of false alarm decreases, the probability of detection will decrease as well. We have choosen to use the Neyman-Pearson criteria: we decide a certain probability of false alarm and we maximize the probability of detection. A different approach is the Bayesian theorem: it minimizes a risk function instead of the P_{FA}, but requires a prior probability of the hypothesis, a so called a priori distribution, in particular it requires the probability of the signal presence, which generally is not known, as in our case. We need to have a test statistic. To do it, we start from the likelihood ratio $L(z), whichis$ the probability of $\boldsymbol{z}$ being a data signal (hypothesis

H_1) and the probability of $\boldsymbol{z}$ being a noise only signal (hypothesis H_0), where $\boldsymbol{z}$ is the vector composed of the samples $z[n]$:

$$L(z) = \frac{p(\boldsymbol{z}; H_1)}{p(\boldsymbol{z}; H_0)} \underset{H_0}{\overset{H_1}{\gtrless}} \gamma$$

When the samples are independent Gaussian variables we can write:

$$L(z) = \frac{\frac{1}{(2\pi\sigma^2)^{\frac{N}{2}}} exp\left[-\frac{\sum_{n=0}^{N-1}(z[n]-A])^2}{2\sigma^2}\right]}{\frac{1}{(2\pi\sigma^2)^{\frac{N}{2}}} exp\left[-\frac{\sum_{n=0}^{N}(z[n])^2}{2\sigma^2}\right]} \underset{H_0}{\overset{H_1}{\gtrless}} \gamma \tag{5.9}$$

By canceling common terms and constants, this relation can be transformed to the test statistics needed to compute a threshold:

$$T(z) = \frac{1}{N}\sum_{n=0}^{N-1} z[n] \underset{H_0}{\overset{H_1}{\gtrless}} \lambda \tag{5.10}$$

Let's notice that we decided to replace γ by λ, because N and σ^2 are all constant factors. What we must do now is analyze the mean value and the variance of $T(z)$, which is our test statistic. After that we can compute the threshold:

$$\begin{aligned} T(z) &\in \mathcal{N}(0, \frac{\sigma^2}{N}) \quad under\ H_0 \\ T(z) &\in \mathcal{N}(A, \frac{\sigma^2}{N}) \quad under\ H_1 \end{aligned}$$

Following [14] , P_{FA} and P_D can be computed using the right-tail probability $Q(\cdot)$, or the probability of exceeding a given value with a Gaussian distribution, as follows:

$$P_D = Q\left(\frac{\lambda - A}{\sqrt{\frac{\sigma^2}{N}}}\right) \tag{5.11}$$

$$P_{FA} = Q\left(\frac{\lambda}{\sqrt{\frac{\sigma^2}{N}}}\right) \tag{5.12}$$

Transforming (5.12) to compute the threshold we obtain:

$$\lambda = \sqrt{\frac{\sigma^2}{N}} Q^{-1} \left(P_{FA}\right) \tag{5.13}$$

where λ is the one threshold that yields the maximum P_D for the given P_{FA}.

5.4 Neyman-Pearson theorem to TR-UWB

The Neyman-Pearson theorem is applied to set a threshold and decide if the signal is present or not, also in the TR-UWB system we are considering in this thesis. First of all, let's define the two hypothesis:

$$\begin{aligned} &Hypothesis\ H_0: \quad \boldsymbol{Z} = \boldsymbol{N} \\ &Hypothesis\ H_1: \quad \boldsymbol{Z} = \boldsymbol{X} + \boldsymbol{N} \end{aligned}$$

where $\boldsymbol{X}$ is the signal model as in (5.6) and $\boldsymbol{N}$ is the noise sample matrix. Actually, the two mentioned hypothesis don't cover the case where the signal is partially present, but we'll take care of this case in the last section of the chapter. Following section 5.1 we know that the elements in $\boldsymbol{N}$ are Independent Gaussian Variables, with null mean value. In the hypothesis H_0 the variance of each noise sample is $\sigma^2_{H_0}$ as in (5.4), whereas in the hypothesis H_1 the variance is $\sigma^2_{H_1}$ as in (5.1). Let's calculate the likelihood ratio $L(z)$:

$$\begin{aligned} L(z) &= \frac{p(\boldsymbol{Z}; H_1)}{p(\boldsymbol{Z}; H_0)} \underset{H_0}{\overset{H_1}{\gtrless}} \gamma \\ &= \frac{\frac{1}{(2\pi\sigma^2_{H_1})^{\frac{N_d L_s}{2}}} exp\left[-\frac{\sum_{n=0}^{N_d L_s}(z[n]-x[n])^2}{2\sigma^2_{H_1}}\right]}{\frac{1}{(2\pi\sigma^2_{H_0})^{\frac{N_d L_s}{2}}} exp\left[-\frac{\sum_{n=0}^{N_d L_s}(z[n])^2}{2\sigma^2_{H_0}}\right]} \underset{H_0}{\overset{H_1}{\gtrless}} \gamma \end{aligned} \tag{5.14}$$

The expression (5.14) is different by (5.9) because in our case the variance of the variables is different in the two different hypothesis. The calculation of the test statistic becomes very complex, but at the end the result is the same as (5.10):

$$T(z) = \frac{1}{N_d L_s} \sum_{n=0}^{N_d L_s - 1} z[n] \underset{H_0}{\overset{H_1}{\gtrless}} \lambda \tag{5.15}$$

where λ depends by both $\sigma^2_{H_0}$, $\sigma^2_{H_1}$ and also by $N_d L_s$. So, the test statistic coincides with the calculation of the mean value of the matrix $\boldsymbol{Z}$. Let's call in general $\hat{\mu}$ the calculated mean value of $\boldsymbol{Z}$: $\hat{\mu} = \frac{1}{N_d L_s} \sum_{n=0}^{N_d L_s - 1} z[n]$. In the hypothesis H_0 we define $\hat{\mu}_{H_0}$ the test statistic:

$$\hat{\mu}_{H_0} = \frac{1}{N_d L_s} \sum_{n=0}^{N_d L_s - 1} z[k+n] = \frac{1}{N_d L_s} \sum_{n=0}^{N_d L_s - 1} n[k+n] \tag{5.16}$$

whereas in the hypothesis H_1:

$$\begin{aligned} \hat{\mu}_{H_1} &= \frac{1}{N_d L_s} \sum_{n=0}^{N_d L_s - 1} z[n] = \\ &= \frac{1}{N_d L_s} \sum_{n=0}^{N_d L_s - 1} (x[k+n] + n[k+n]) = \frac{1}{N_d L_s} \sum_{n=0}^{N_d L_s - 1} x[k+n] + \frac{1}{N_d L_s} \sum_{n=0}^{N_d L_s - 1} n[k+n] \\ &= \frac{N_f E_h}{L_s} + \frac{1}{N_d L_s} \sum_{n=0}^{N_d L_s - 1} n[k+n] \end{aligned} \tag{5.17}$$

To write (5.17) we used the result of (5.7). In both the hypothesis the test statistic is the sum of $N_d L_s$ independent gaussian variable and this means that it still is a gaussian variable. $\hat{\mu}_{H_0}$ is a gaussian variable with null mean value and variance equal to

$$\sigma^2_{\mu_{H_0}} = \frac{1}{N_d L_s} \sigma^2_{H_0} \tag{5.18}$$

See figure 5.4
$\hat{\mu}_{H_1}$ is a gaussian variable with mean value

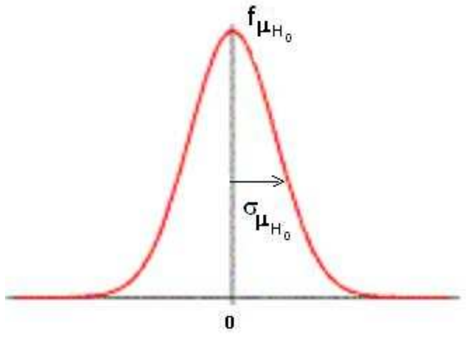

Figure 5.4: Probability density function of $\hat{\mu}_{H_0}$

$$\mu = \frac{N_f E_h}{L_s} \tag{5.19}$$

and variance

$$\sigma^2_{\mu_{H_1}} = \frac{1}{N_d L_s}\sigma^2_{H_1} \tag{5.20}$$

See figure 5.5

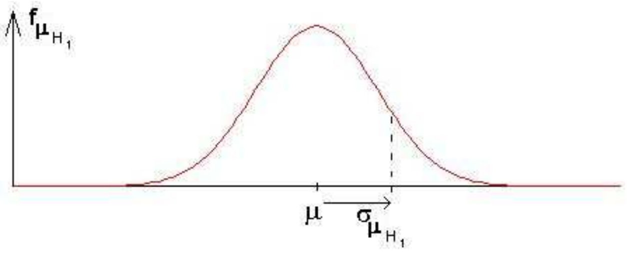

Figure 5.5: Probability density function of $\hat{\mu}_{H_1}$

The first clarification to do is that, really, to have mean value of $\hat{\mu}_{H_1}$ equal to $\frac{N_f E_h}{L_s}$ the detection training sequence must be composed by $N_d + 1$ symbols, because in the first column of the matrix $\boldsymbol{X}$ must be present the tail of the preceding symbol: only in that case it's true that, in the equation (5.6), the matrix $\boldsymbol{C}_\tau$ has the structure shown in figure 3.5 So let's start finally to understand how we can detect the presence of the signal. Let's suppose in the beginning the signal is not present. Let's suppose that when the receiver is turned on, the user doesn't transmit for a certain

fixed period,at least. Under this assumption we can estimate the variance of the variable $\hat{\mu}_{H_0}$ by estimating the variance of the noise samples applying the maximum likelihood criteria:
$\hat{\sigma}^2_{H_0} = \frac{1}{N_l}\sum_{i=1}^{N_l} n^2[i]$, where N_l depends on how long we decide the user doesn't transmit, at least, since the receiver has been turned on. As N_l is larger, the estimation of the variance is better. Then, we begin to collect the samples in the matrix $\boldsymbol{Z}$ and we begin to calculate the test statistic $\hat{\mu}$:

$$\hat{\mu} = \frac{1}{N_d L_s}\sum_{i=1}^{N_d L_s} z[i] \tag{5.21}$$

When we have the estimation of the mean value we must compare it with a threshold λ. If $\hat{\mu}$ is bigger than λ we can assert that the signal is present, otherwise the signal is not present and we must rebuild the matrix $\boldsymbol{Z}$ removing the first column and adding another column with the L_s following samples. Then we re-estimate the mean value and we compare it with the threshold. We repeat the procedure until the estimation becomes bigger than the threshold. In that case we have detected the presence of the signal. Now we have to understand how to set the threshold λ. As we said in the preceding section, we'll calculate the threshold under a given P_{FA}: equation (5.13). We already know the statistics of our decision variable in the hypothesis H_0 and, applying (5.1) as we'll show later, we know the statistics in the hypothesis H_1 too. Nevertheless between these two cases there is a transition, during which the probability density function of the variable $\hat{\mu}$ changes from the first to the second (from the pdf of $\hat{\mu}_{H_0}$ to the pdf of $\hat{\mu}_{H_1}$). See figure 5.6

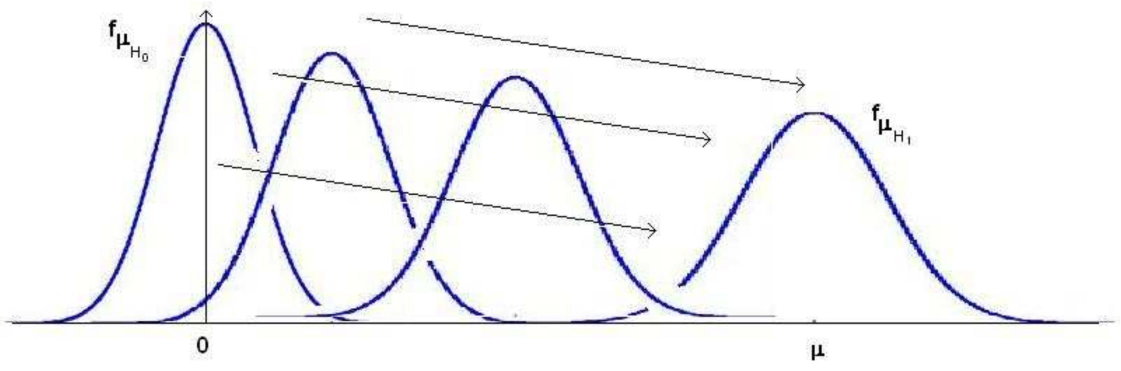

Figure 5.6: Transition of the PDF of $\hat{\mu}$

The event False Alarm occurs when, in the hypothesis H_0, the decision variable is bigger than the threshold λ. The threshold is set in a way that the False Alarm Probability (P_{FA}) is as small as we want.

$$P_{FA} = P_r[(\hat{\mu}|H_0) > \lambda] = P_r[\hat{\mu}_{H_0} > \lambda] = \int_{\lambda}^{+\infty} \boldsymbol{f}_{\mu_{H_0}}(x)dx \tag{5.22}$$

The integration in (5.22) is the integration of a Gaussian Variable: see figure 5.7

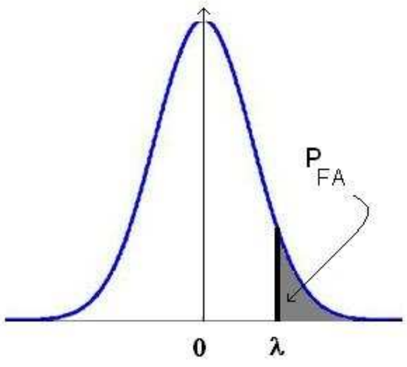

Figure 5.7: False Alarm probability, depending on the threshold

So, as we know, we can write:

$$P_{FA} = Q\left(\frac{\lambda}{\sqrt{\sigma^2_{\mu_{H_0}}}}\right) \tag{5.23}$$

And then:

$$\lambda = \sqrt{\sigma^2_{\mu_{H_0}}}Q^{-1}(P_{FA}) \tag{5.24}$$

That is the same result as (5.13) We have already said how to estimate the variance $\sigma^2_{\mu_{H_0}}$, so, we set the threshold deciding the P_{FA} and applying the formula (5.24). When the signal begins to be received, the matrix $\boldsymbol{Z}$ begins to include also data samples, starting by the last column and the probability density function of $\hat{\mu}$ begins to move towards the pdf of $\hat{\mu}_{H_1}$. So, intuitively, the probability that $\hat{\mu}$ goes over the threshold begins to grow. What we can calculate is the Detection Probability P_D in the hypothesis H_1, when the pdf of our test statistic coincides with the pdf of

$\hat{\mu}_{H_1}$. It will depend on the threshold that we had set and also on the power of the received signal.

$$P_D = P_r[(\hat{\mu}|H_1) > \lambda] = P_r[\hat{\mu}_{H_1} > \lambda] = \int_{\lambda}^{+\infty} \boldsymbol{f}_{\mu_{H_1}}(x)dx \tag{5.25}$$

As before,the integration in (5.25) is the integration of a Gaussian variable: see figure 5.8

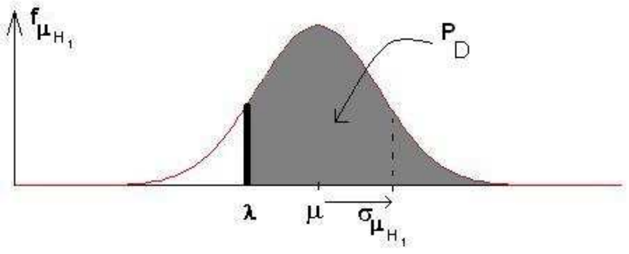

Figure 5.8: Detection probability, depending on the threshold and the power of the signal, included in both μ and $\sigma_{\mu_{H_1}}$

So, we can write:

$$\begin{aligned} P_D &= Q\left(\frac{\lambda-\mu}{\sqrt{\sigma^2_{\mu_{H_1}}}}\right) \\ &= 1-Q\left(\frac{\mu-\lambda}{\sqrt{\sigma^2_{\mu_{H_1}}}}\right) \end{aligned} \tag{5.26}$$

Let's try to understand how the expression (5.26) depends on the power of the signal. First of all we can replace λ with the expression (5.25). Then, let's remember the expression of μ (5.19) and of $\sigma^2_{\mu_{H_1}}$. In particular, for this last one, remember the relation between (5.20) and (5.1):

$$\sigma^2_{\mu_{H_1}} = \frac{1}{N_dL_s}\sigma^2_{H_1} = \frac{1}{N_dL_s}4T_{sam}(1+\frac{S}{N})N_0^2\beta \tag{5.27}$$

N_f , L_s , T_w are the parameters that we have decided designing the system. E_h depends on the channel, in fact, remembering (5.8) and (2.3):

$$E_h = \sum_{n=0}^{P_h-1} h(n) = \sum_{n=0}^{P_h-1} \int_{nT_{sam}}^{(n+1)T_{sam}} h^2(t)dt = \int_0^{T_h} h^2(t)dt \tag{5.28}$$

where T_h is the length of the UWB physical channel.
But what is $h(t)$?
$h(t)$ is not only the channel. From (2.1) we realize that $h(t)$ is the convolution between the transmitted pulse $g(t)$, the communication channel $h_p(t)$ and also the antenna response $a(t)$. So, E_h is right the power of the signal at the input of the pulse pair correlator. In other words, $E_h = S$. Remembering also the expression (5.1) let's rewrite the Detection Probability:

$$P_D = 1 - Q\left(\frac{\frac{N_f S}{L_s} - \sqrt{\hat{\sigma}^2_{\mu_{H_0}}} Q^{-1}(P_{FA})}{\sqrt{4T_{sam}(1+\frac{S}{N})N_0^2\beta/N_d L_s}}\right) \tag{5.29}$$

The expression (5.29) is the Detection Probability we have applying this detection algorithm. Let's make some comments. First of all, as N_d gets larger, P_D also increases. Then, after having decided P_{FA}, we can increase the P_D also increasing the power of the signal (so, increasing S). The problem is that we cannot calculate the exact expression of P_D, because of the simple reason that we don't know the channel, that means we don't know $h_p(t)$ and so we can't calculate S. Nevertheless. there is a case where we can calculate at least the minimum Detection Probability ($P_{D_{min}}$). This case occurs when we are sure that there is the Line Of Sight (LOS) between the transmitter and the receiver. In fact, in this case, we can calculate the attenuation of the direct ray and so we can consider only that ray to calculate the minimum value of S:

$$S_{min} = K \int_0^{T_h} g(t) \star a(t)dt \tag{5.30}$$

where K is the attenuation due to the distance between the transmitter and the receiver: we'll consider the maximum possible distance. Let's notice that we can know also N by the estimation of $\sigma^2_{\mu_{H_0}}$. In fact, the expression (5.1) allows us to calculate N using $\widehat{\sigma^2_{\mu_{H_0}}}$ and inverting the formula (5.4). Anyway it's simple to understand that, even in the case of the presence of the LOS, we would like to have a channel with strong multipath, because we would like to take advantage of all the paths, in fact all the paths contribute to increase the signal power S.

5.5 The complete training sequence

As we said till now, to detect the signal we send a training sequence, called detection training sequence, composed of $N_d + 1$ symbols, in which the chips of the code sequence are all equal to 1. When the decision variable $\hat{\mu}$ goes over the threshold we say to have detected the signal and we apply the synchronization algorithm. Let's notice that the synchronization algorithm is applied to a matrix build by collecting symbols in which the code sequence doesn't have all chips equal to 1. It means that the detection algorithm and the synchronization algorithm are applied to two different parts of the training sequence. In fact, we have divided the training sequence in two parts: Let's call the first part "detection training sequence" and the second "synchronization training sequence". Let's notice that in the beginning the samples are only noise samples and the probability to go over the threshold is P_{FA}. Then, when the signal begins to arrive, the probability of detection grows. This happens when we begin to receive the detection training sequence. What could happen is that the decision variable goes over the threshold before the detection training sequence is completely received. This means that if we apply the synchronization algorithm immediately after the detection we are wrong because the data matrix that we use to make the synchronization would be composed in part also by the symbols belonging to the detection training sequence. So, what we must do is to be sure that after the detection we apply the synchronization algorithm only to the synchronization training sequence. To do this, the entire training sequence is so structured: the first part, that is the detection training sequence is composed by $N_d + 1$ symbols; the second part, that is the synchronization training sequence is composed by $N_d + N_s + 1$ symbols, where N_s is the number of columns of the data matrix that we build to apply the synchronization algorithm. So, when we detect the signal, we jump the following N_d symbols and we apply the synchronization algorithm. Only in this way we are sure that we don't include in the synchronization algorithm symbols belonging to the detection training sequence. The last thing to notice is that in the case of multi-users, every users have the same detection training sequence. This means that when one user transmits, everybody could detect the presence of a signal, but after the synchronization and demodulation they will recognize that the signal is not for them. Only the interested user will recognize his code sequence.

5.5.1 An alternative approach

There is another simple way to detect the presence of the signal. Instead of dividing the training sequence in two parts, we send only the synchronization training sequence. The idea is simply to apply the synchronization algorithm even if we don't know if the signal is present or not. Then, remembering formula (3.16), we compare the maximum value we find (for each value of τ) with a threshold, set following a certain criteria as still the Neyman-Pearson theorem. If the maximum value is bigger than the threshold we say that we have received the signal and in the same moment we have already estimated the offset τ. Otherwise we say that we received only noise. The problem of this approach is that we apply the synchronization algorithm even if we don't receive the signal: every L_s samples we must do a FFT operation, then the multiplication by matrix $\tilde{\boldsymbol{C}}_{inv}$ and after that an IFFT operation. These are a lot of calculations and it's also possible that they are completely useless, since it could happen that we won't ever receive the signal.

Chapter 6

Future developments

In the CAS group at TU-Delft, the synchronization algorithm is being improved. In their last article [12], Yiyin Wang, Geert Leus and Alle-Jan van der Veen changed slightly the algorithm: it's based on the same concepts explained in the preceding chapters, but the last modifications make it be able to handle ISI too and to achieve joint channel and timing estimations, keeping a low complexity, due to property of the circulant matrix in the data model.
Three channel estimators and three equalizers are derived. It's interesting to take a look to the simulations presented in [12] regarding the estimation of the delay in the synchronization algorithm:

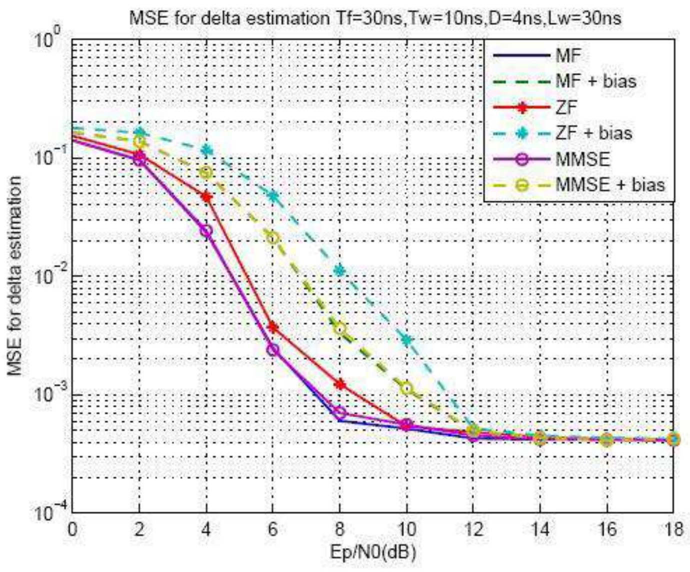

Figure 6.1: MSE performance for the estimation of the delay with $L_w = 30ns$

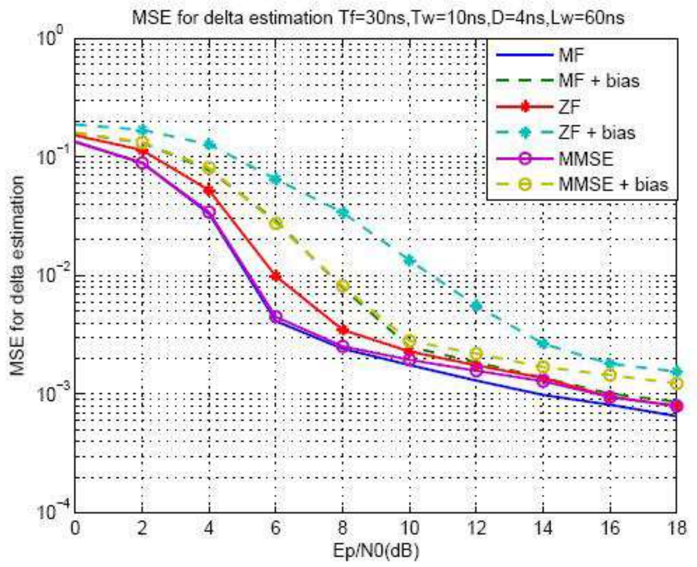

Figure 6.2: MSE performance for the estimation of the delay with $L_w = 60ns$

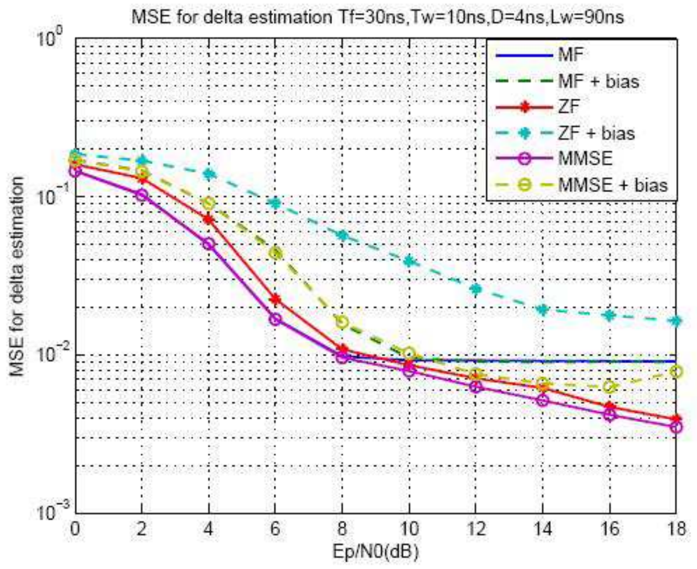

Figure 6.3: MSE performance for the estimation of the delay with $L_w = 90ns$

The figures 6.1, 6.2 and 6.3 are taken from [12] and "delta" indicates the delay τ, whereas T_w is T_{sam}.
The bias caused by Inter Pulse Interference (IPI) is also considered, whereas it wasn't till now. Let's put our attention to the curves related to the performance of the matched filter (MF) and the synchronization algorithm in its first version

(MMSE). As we can see the performances are practically the same. Nevertheless, this simulations were done using IEEE UWB channel model CM3 [15], in which the channel attenuates exponentially in a way that in one frame there is most of the energy. It means that in these simulations the IFI is not so serious. The MMSE estimator is able to handle more serious IFI and ISI, compared with the matched filter.

Conclusions

In this thesis a new synchronization algorithm for UWB systems has been presented. The property of the circulant matrix makes the complexity low, more or less the same of the traditional matched filter. The performances are still good, but the new algorithm can handle lots interferences and it could be improved to allow higher data rate communications, for example making the data model more accurate. The challenge is in fact to reach high data rate in UWB systems.

Bibliography

[1] *"FCC notice of proposed rule making, revision of part 15 of the commision's rules regarding ultra-wideband transmission systems".* Tech. Rep. ET-Docket 98-153, Federal Communications Commission, Washington,D.C., Apr. 2002.

[2] *"Harmonise radio spectrum usefor ultra-wideband systems in the european union".* Tech. Rep. European Electronic Communications Committee, Copenhagen, Denmark, Mar. 2005.

[3] L.Yang and G.B. Giannakis: *"Ultra-wideband communications: An idea whose time has come".* IEEE Signal Process. Mag., vol.21, pp. 26-54, Nov. 2004

[4] D. Cassioli, M. Win and A. Molisch: *"The ultra-wideband bandwidth indoor channel: From statistical model to simulations".* IEEE Journal on Selected Areas in Communications, vol. 20, pp. 1247-1257, Aug. 2002.

[5] J.Foerster and Q. Li *"Uwb channel modeling contribution from intel".* Tech. Rep., IEEE P802.15 Working Group for Wireless Personal Area Networks (WPANs), 2003

[6] R. Hoctor and H. Tomlinson: *"Delay-hopped transmitted reference RF communications".* IEEE Conference on Ultra Wideband Systems and Technologies, pp. 265-270,2002.

[7] J. Choi and W. Stark: *"Performance of ultra-wideband communications with suboptimal receivers in multipath channels".* IEEE Journal on Selected Areas in Communications, vol. 20, pp. 1754–1766, Dec. 2002.

[8] Andreas Schranzhofer: *"Acquisition for a Transmitted Reference UWB Receiver".* Master's thesis, May 2007.

[9] S. Aedudodla, S. Vijayakunmaran and T.F. Wong: *"Timing acquisition in ultra-wideband communication system"*. IEEE Trans. Veh. Technol., vol. 54, pp. 1570-1583, Sept. 2005.

[10] R. Djapic, G. Leus and A.J. van deer Veen: *"Blind synchronization in asynchronous multiuser uwb networks based on the transmit-reference scheme"*. in Proc. IEEE Asilomar Conf. Signals, Systems and Computers, Pacific Grove, CA, Nov. 2004, vol.2, pp. 1506-1510

[11] Q.H. Dang and A.J. van der Veen: *"Signal processing model and receiver algorithms for a higher rate multi-user TR-UWB system"*. In Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Proc. (ICASSP 07), Honolulu (HI), IEEE, pp. III.581-584, April 2007.

[12] Y. Wang, G. Leus and A.J. van der Veen: *"Channel Estimation fot Transmitted Reference Ultra-Wideband Communication Systems"*, submitted to IEEE International Conference on Ultra Wideband, Sept. 2008

[13] Ralph T.Hoctor and Harold W. Tomlinson: *"An Overview of Delay- Hopped, Transimitted-Reference RF Communications."* GE Research & Development Center, 2001CRD198,January 2002

[14] S. M. Kay: *"Fundamentals of Statistical Signal Processing - Detection Theory"*. ser. Prentice Hall signal processing series. Prentice Hall PTR, 1993, vol. II.

[15] J.R. Foerster: *"Channel modelling sub-committe report final"*. Tech. Rep. IEEE P802.15-02/368r5-SG3a, IEEE P802.15 Working Group for WPAN, 2002

Thanks

I would like to express my sincere appreciation and gratitude to Prof. dr. ir. Alle-Jan van der Veen, my supervisor at TU-Delft, who has followed my project giving me important suggestions and teaching me the right approach to make a good work.

Thanks to Ing. Marco Moretti, my supervisor in Pisa, for giving me this opportunity and helping me in hard moments.

Thanks also to Yiyin Wang, PhD student at TU-Delft, for her indispensable advices and her kindness.

Special thanks to my family, that trusted me and gave me a never-ending support.

A special hug to my two friends Claudio and Giulio with whom I spent the years of university enjoying and helping each others.

Heartfelt thanks to my friends of my city that made me not to feel alone even if I was far away.

I will never forget my friends in Delft, who were always close to me during this important experience.