Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Signal and Information Processing Laboratory
Institut für Signal- und Informationsverarbeitung

Winter Semester 2002/2003 Prof. Dr. H.-A. Loeliger

Semester Project

Continuous-Time Synchronization

Tobias Koch

Advisor: Justin Dauwels

Co-Advisors: Matthias Frey and Patrick Merkli

aufgabenstellung

Acknowledgments

This project came into existence with friendly assistance of many people:

First of all, many thanks to Justin, Matthias and Patrick for their kind help and for supporting me at any time during the last seven weeks.

I am also grateful to Prof. Dr. H.-A. Loeliger for the interesting meetings we had and for brimming over with enthusiasm for the topic of my project. It was a pleasure for me to work here during the last seven weeks!

Zurich, March 28, 2003

Tobias Koch

Abstract

We design a novel low-complexity pseudonoise (PN) sequence synchronizer referred to as Noise Lock Loop (NLL). The NLL achieves synchronization by estimating the probabilities of the bits in the PN sequence.

The NLL is interesting for several reasons. In classical spread spectrum communication systems, PN synchronization is separated into two phases: There is an initial acquisition phase and a tracking phase after the signal has been acquired. Acquisition and tracking are typically performed by two separate synchronization systems. The NLL is a single system of low-complexity that both acquires and tracks noisy PN-sequences. Furthermore, classical PN synchronizers are clocked. We develop both clocked and unclocked NLLs and study their behavior by simulation.

We proceed in two steps. First we a consider a clocked PN communication system. It consists of a Linear Feedback Shift Register (LFSR) whose sequences are transmitted over an additive white Gaussian noise (AWGN) channel and are fed into a clocked NLL subsequently. In this case the communication system operates in discrete time. The unit of time is determined by the clocks in the LFSR and the NLL, which are assumed to be perfectly aligned. We refer to this operation mode as discrete-time synchronization.

Then two different unclocked PN communication systems are considered. In the first system LFSR sequences are transmitted, in the other system one transmits arbitrary PN sequences. In both systems the sequences are corrupted by AWGN and received by an unclocked NLL. In this case the communication system operates so as to say in continuous time. The system does not contain clocks. We refer to this operation mode as continuous-time synchronization.

For the discrete-time system the *Probability of Synchronization* $P_{Synch}(k)$ and the *Time Until Acquisition* (TUA) are defined as a performance measure. The performance of both continuous-time systems is measured by the Mean-Square-Error computed between the signals in the transmitter and the signals in the receiver.

We summarize our results as follows. The simulations of the discrete-time system show that $P_{Synch}(k)$ tends asymptotically to a value $P_0 \leq 1$.

The lower the signal-to-noise ratio (SNR), the smaller P_0. In addition, the rise of $P_{Synch}(k)$ depends on the SNR as well as on the length of the LFSR: the lower the SNR or the longer the LFSR, the smaller the rise. The same holds for the TUA, i.e. the lower the SNR or the larger the LFSR length, the larger the TUA.

The simulations of the continuous-time system show that the MSE decreases monotonically as a function of time until it reaches a value $\epsilon \geq 0$. The lower the signal-to-noise ratio (SNR), the larger ϵ becomes and the slower the MSE decreases.

The NLL is just in its infancy, especially the unclocked NLL. However, we have shown that the design of both clocked and unclocked NLLs is theoretically possible. The natural next step is the implemention of such systems in hardware.

Contents

List of Figures

Chapter 1

Introduction

Nowadays spread spectrum signals are often used to transmit information over a communication channel at a given rate. Spread spectrum signals are characterized by the fact that their bandwidth is much greater than necessary to achieve this information rate. This large redundancy is inherent in spread spectrum signals and in consequence these signals are robust against interference.

One type of spread spectrum signals are *pseudorandom* signals, also referred to as *pseudonoise* (PN) signals. The best known binary PN sequences are the *maximum-length shift-register sequences* generated with a shift register with feedback as shown in Figure 1.1. We discuss the generation of *PN sequences* here not in more detail. We refer to [1] for more information.

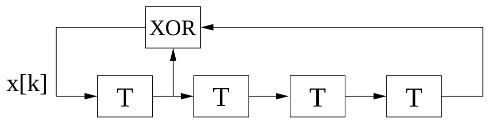

Figure 1.1: Shift register with feedback

Pseudorandom signals are similar to random noise and, therefore, difficult to demodulate by receivers other than the intended ones. Due to this possible message privacy spread sprectrum signals are interesting in multiple-user communication systems in which a number of users share a common channel.

Figure 1.2 shows a block diagram of a pseudorandom spread spectrum communication system. It consists of a *modulator*, a *demodulator* and two *pseudorandom pattern generators.*

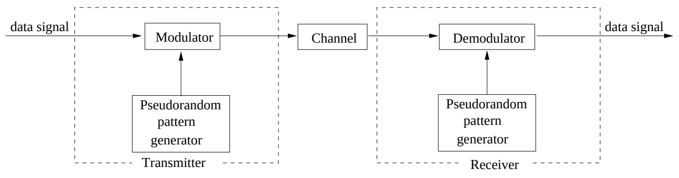

Figure 1.2: Block diagram of a spread spectrum communication system

The modulator impresses the pseudorandom sequence, generated by the pseudorandom pattern generator, on the transmitted information bearing signal and maps then the digital information onto analog waveforms that match the characteristics of the channel. In order to recover the data the receiver needs to know the PN sequence generated in the transmitter.

In a typical spread spectrum modulation scheme, referred to as direct sequence modulation, the binary data signal is multiplied with the PN sequence $p_k(t)$:

$$p_k(t) = (1 - 2x[k])p(t - iT_c),$$

where $x[k]$ is the output of the shift register shown in Figure 1.1 with elements $\{0, 1\}$, $p(x)$ is a pulse shape and T_c is called the the chip interval. The bandwidth W of the PN sequence $p_k(t)$ is given by $T_c = 1/W$.

In order to recover the data bits, the receiver needs to multiply the received sequence with the proper PN sequence. This PN sequence needs to be aligned to the received signal. This alignment, referred to as PN synchronization, is achieved in two phases: The *acquisition* and the *tracking phase*. During PN synchronization, the data signal is held constant.

In the acquisition phase, the receiver needs to decide on the PN sequence generated in the transmitter. Acquisition is achieved when the PN sequences generated in the transmitter and in the receiver are identical and time-synchronized to within one half chip interval T_c.

In the tracking phase the receiver aligns its internal PN sequence further to the incoming signal until no timing offset remains.

In this project we design a novel low-complexity PN sequence synchronizer referred to as Noise Lock Loop (NLL). The NLL achieves synchronization by estimating the probabilities of the bits in the PN sequence.

The NLL is interesting for two reasons. As stated in the above, in classical spread spectrum communication systems, PN synchronization is separated into two phases: There is an initial acquisition phase and a tracking phase after the signal has been acquired. Acquisition and tracking are typically performed by two separate synchronization systems. The NLL is a single system of low-complexity that both acquires and tracks noisy PN sequences. Furthermore, classical PN synchronizers are clocked. We develop both clocked and unclocked NLLs and study their behavior by simulation.

We proceed in two steps. First we a consider a clocked PN communication system. It consists of an LFSR whose sequences are transmitted over an additive white Gaussian noise (AWGN) channel and are fed into a clocked NLL subsequently. In this case the communication system operates in discrete time. The unit of time is determined by the clocks in the LFSR and the NLL, which are assumed to be perfectly aligned. We refer to this operation mode as discrete-time synchronization.

Then two different unclocked PN communication systems are considered. In the first system LFSR sequences are transmitted, in the other system one transmits arbitrary PN-sequences. In both systems the sequences are corrupted by AWGN and received by an unclocked NLL. In this case the communication system operates so as to say in continuous-time. The system does not contain clocks. We refer to this operation mode as continuous-time synchronization.

This report is organized as follows: In Chapter 2, the discrete-time communication system is considered. In Chapter 3, we investigate the continuous-time system. We conclude this report with Chapter 4 where an overview of this project is shown and a summary of the results is presented.

Chapter 2

Discrete-Time Synchronization

We investigate a discrete-time communication system shown in Figure 2.1. This system consists of a *Linear Feedback Shift Register* (LFSR) as a transmitter and the *Noise-Lock Loop* (NLL) as a receiver. The in the LFSR generated *PN sequence* is transmitted over an *Additive White Gaussian Noise* (AWGN) channel using *Binary Phase Shift Keying* (BPSK) modulation. The AWGN channel is characterized by its variance σ^2.

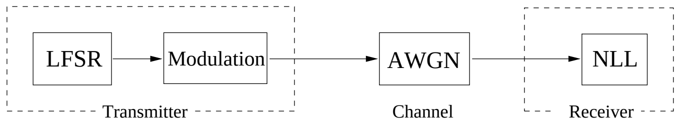

Figure 2.1: Discrete-time communication system

In a discrete-time system the transmitter and the receiver are time-synchronized. Therefore the job of the receiver is to estimate the transmitted *PN sequence* $x[k]$ given the noisy observation $y[k]$.

In order to study this communication system, we perform a simulation for several SNRs (i.e. $\sigma-$values) and different LFSR lengths. In order to measure the performance of our system we define the *Probability of Synchronization* and the *Time Until Acquisition* (TUA).

We investigate this discrete-time communication system as follows: In Section 2.1 we consider the LFSR. Section 2.2 is about the NLL that we use as a receiver. Then results from the simulation are presented and discussed.

2.1 The Linear Feedback Shift Register

In order to generate PN sequences we use a Linear Feedback Shift Register (LFSR) that consists of a cascade of delay cells (a so called shift register) and a logic XOR.

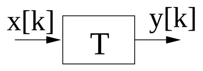

Figure 2.2: Delay cell

The delay cell shown in Figure 2.2 stores the actual value at the input $x[k]$[1] and repeats it after one time step, so the output $y[k]$ can be written as

$$y[k] = x[k-1]. \tag{2.1}$$

We define the length of an LFSR as the number of delay cells that are in the register, so an LFSR with L delay cells has length L.

The input of the first delay cell $x[k]$ in the LFSR depends on the signals in the registers such that

$$x[k] = x[k-M] \oplus x[k-N] \tag{2.2}$$

where $\oplus$ denotes the logic XOR.

Note that if N is not equal to the LFSR length L then we can substitute this LFSR with one that has length N, since the signals $x[k-i]\ \forall\, i > N$ have no influence on $x[k]$. So in the following we will assume without loss of generality that N is equal to L.

Figure 2.3 shows an LFSR with $M = 1$ and length $L = 4$. Here the input of the first delay cell is equal to

$$x[k] = x[k-1] \oplus x[k-4]. \tag{2.3}$$

In an LFSR of length L there are 2^L different combinations of the signals $x[k-1], x[k-2], \ldots, x[k-L]$ possible. We call these combinations the *states* of the LFSR.

The signal $x[k]$ depends on M as well as on the initial state. If the initial state is the *all-zero state* with $x[k-1] = x[k-2] = \cdots = x[k-L] = 0$, the output $x[k]$ will be the *zero sequence* ($x[k] = 0\ \forall k$) since the next input of

[1] $x[k]$ denotes a discrete-time signal

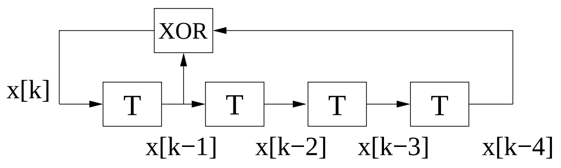

Figure 2.3: Linear Feedback Shift Register of length 4

the register will be a zero and so the LFSR stays in the all-zero state. Note that this holds independent of the choice of M and L. The zero sequence is no PN sequence. In order to generate a PN sequence we need to initialize the LFSR with a state different to the all-zero state.

If we compute the state evolution of an LFSR, after at most $2^L - 1$ iterations we return to the initial state. This evolution can be represented by a circle. Figure 2.4 shows such circles for the LFSR in Figure 2.3. One circle corresponds to the all-zero state. The second circle shows the state evolution, if the initial state is different to the all-zero state. For example, if the state is 0001, the output of the XOR is 1, which is fed into the register. The next state is therefore 1000. After fifteen steps the LFSR returns to the initial state. Since the signal $x[k]$ depends on the state in the LFSR, the output sequence is periodic with a period of fifteen time steps. Note that this is the maximum period of a PN sequence that can be generated by an LFSR of length four. In general the maximum period of a PN sequence generated by an LFSR of length L is $2^L - 1$.

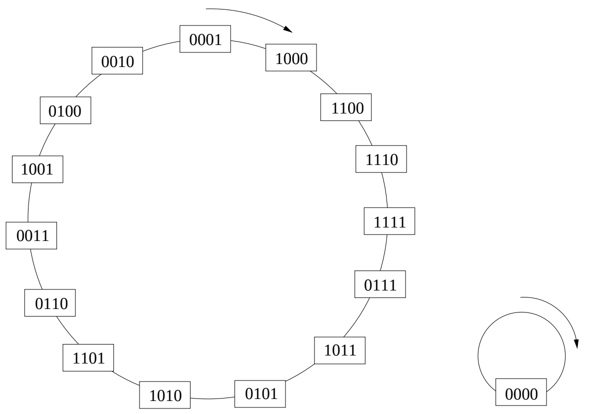

Figure 2.4: State evolution for an LFSR with $x[k] = x[k-1] \oplus x[k-4]$

The circles shown in Figure 2.5 were created by an LFSR defined by

$x[k] = x[k-2] \oplus x[k-4]$. Contrary to the state evolution in Figure 2.4 we have four circles. The bottom right circle is the circle corresponding to the all-zero state. The generated PN sequences have a period of 6 time steps (top left and top right) and 3 time steps (bottom left), which is smaller than 15 and therefore not the maximum period. Since the number of possible states is equal to 2^L, the LFSR creating the state evolution with the fewest circles is able to generate a PN sequence of maximum period. It happens to be the LFSR defined by $x[k] = x[k-1] \oplus x[k-L]$ that creates only two circles in the state evolution and is therefore able to generate a PN sequence of maximum period (note that one circle is always the circle corresponding to the all-zero state and, therefore, two is the minimum number of circles).

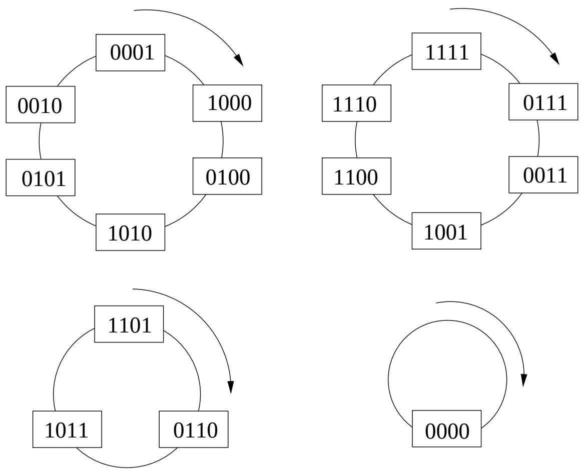

Figure 2.5: State evolution for an LFSR with $x[k] = x[k-2] \oplus x[k-4]$

As mentioned before, the output of the LFSR is modulated using BPSK. The sequence of logic bits $x[k] \in \{0, 1\}$ is mapped to a signal $\check{x}[k] \in \{+1, -1\}$, where a logic 0 is mapped to 1 and a logic 1 to -1.

The examination of the LFSR is here finished. In the next section we consider the Noise-Lock loop that we use as receiver.

2.2 The Noise-Lock Loop

The task of the receiver is to decide on the state in the transmitter based on the received signal $y[k]$. If the receiver does a *maximum a posteriori (MAP) decision* it is optimal, since it decides on the most likely state given the observation $y[k]$ and, therefore, minimizes the probability of an error [2].

A MAP decision can be done with a trellis, where the most likely state is determined by using the Viterbi algorithm [3]. Figure 2.6 shows one trellis section for an LFSR defined in (2.3). Note that from every state there is only one transition to a next state allowed since the state evolution is deterministic, so the receiver only needs to decide on the initial state. However, for an LFSR of length L, $2^L - 1$ possible paths exist through the trellis (if we assume that the zero sequence is not allowed) and so the computational demand of the Viterbi algorithm grows exponentially in L (for $L = 16$ there exist $65'535$ possible paths, for $L = 32$ there are more than $4 \cdot 10^9$ paths).

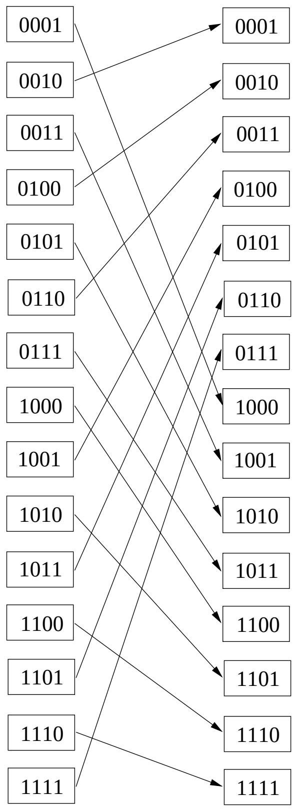

Figure 2.6: Trellis section for a LFSR of length 4

Therefore, we propose a less complex receiver: The Noise-Lock loop (NLL) shown in Figure 2.7. It estimates the a posteriori probability of the state in the LFSR given the noisy observation $y[k]$.

The NLL consists of a cascade of delay cells, a *Soft-Equal* and a *Soft-XOR gate* and the block $p(y|x)$ that translates the signal $y[k]$ into a probability.

In the following we will explain these components.

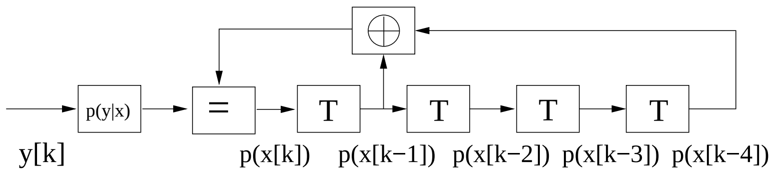

Figure 2.7: Noise-Lock Loop with NLL length 4

Contrary to the LFSR in Section 2.1 the NLL works with probabilities instead of bit sequences. The translation from the signal $y[k]$ into a probabilty is done by the block $p(y|x)$. According to *Bayes' Rule*, we can write the probability of $x[k]$ given $y[k]$ as

$$p(x|y) = \frac{f(y|x)p(x)}{f(y)} = \gamma f(y|x) \tag{2.4}$$

where the last equality follows by the fact, that $p(X=1) = p(X=0)$. Note that $p(x)$ denotes a probability and $f(y)$ a probability density function. γ is chosen so that $p(X=1|y) + p(X=0|y) = 1$. Since the PN sequence is transmitted over an AWGN channel, $f(y|x)$ is a Gaussian distribution.

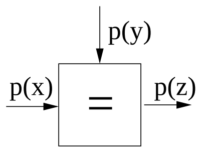

Figure 2.8: Soft-Equal gate

The component shown in Figure 2.8 is called the Soft-Equal gate and computes the probability $p(z)$ given $p(x), p(y)$ and the condtion $z = y = x$:

$$\begin{aligned} p(Z=0) &= \beta p(X=0)p(Y=0) \\ p(Z=1) &= \beta p(X=1)p(Y=1) \end{aligned} \tag{2.5}$$

where β is a scale-factor to ensure that $p(Z=0) + p(Z=1) = 1$.

Figure 2.9 shows the Soft-XOR gate. It computes the probability $p(z)$ given $p(x), p(y)$ and the condition $z = x \oplus y$, where $\oplus$ denotes the logic XOR:

$$\begin{aligned} p(Z=0) &= p(X=0)p(Y=0) + p(X=1)p(Y=1) \\ p(z=1) &= p(X=0)p(Y=1) + p(X=1)p(Y=0). \end{aligned} \tag{2.6}$$

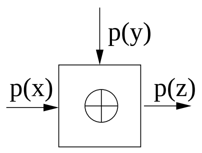

Figure 2.9: Soft-XOR gate

The delay cell in the NLL delays the probability at the input $p\left(x[k]\right)$ by one time step. The output $p\left(z[k]\right)$ of the delay cell can be written as

$$p\left(z[k]\right) = p\left(x[k-1]\right) \tag{2.7}$$

In the LFSR the input is binary, whereas in the NLL the input is a bit probability namely the probability of the corresponding binary signal in the LFSR. We define the length L of the NLL as the number of delay cells.

By comparing the Noise-Lock loop shown in Figure 2.7 to the LFSR shown in Figure 2.3, it can be seen that except the Soft-Equal gate and the block $p(y|x)$ the components of the NLL are a soft-version of the components in the LFSR.

An NLL that is able to synchronize to a PN sequence generated with an LFSR needs to have the same structure, i.e. $M_{LFSR} = M_{NLL}$ and $L_{LFSR} = L_{NLL}$.

The input of the register $p\left(x[k]\right)$ depends on the contents of the register and, different from the LFSR, on the observation $y[k]$. We can write it as

$$\begin{aligned} p\left(x[k]\right) &= \Big[p\left(x[k-M]\right) \bigoplus p\left(x[k-L]\right)\Big] \boxed{=}\, p\left(x[k] \,\big|\, y[k]\right) \\ &= p\big(x[k] = (x[k-M] \oplus x[k-L]) = y[k]\big) \end{aligned} \tag{2.8}$$

where $\bigoplus$ denotes the *Soft-XOR* and $\boxed{=}$ the *Soft-Equal*. The second equality follows by (2.5) and (2.6).

The NLL computes the bit probabilities of the signals $x[k], x[k-1], \ldots, x[k-L]$ in the LFSR given the observation $y[k]$ and is therefore a soft-version of the LFSR in Section 2.1.

Contrary to the trellis, the NLL considers only the probabilities $p\left(x[k-M]\right)$ and $p\left(x[k-L]\right)$ and is therefore sub-optimal. Note that even if the NLL length L goes to infinity the NLL considers only these two probabilities, so it is not possible to improve the receiver by increasing the NLL length.

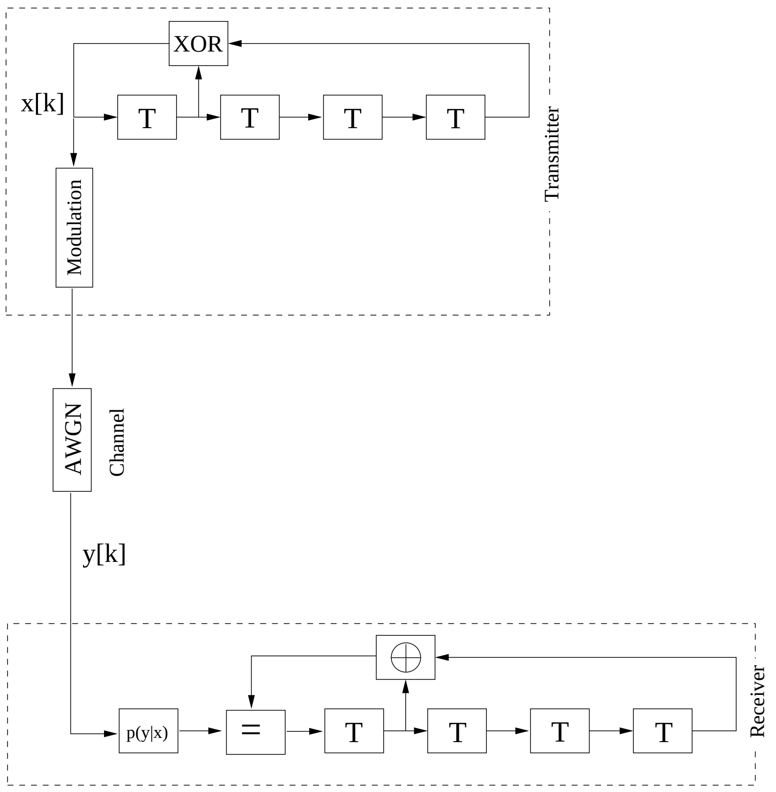

Figure 2.10: Discrete-time communication system using an LFSR of length 4 as a transmitter and an NLL as a receiver

With the Linear Feedback Shift Register and the Noise-Lock loop we investigated all components of the communication system. In the next section we present results from the simulations we performed.

2.3 Results

In order to examine the behavior of the discrete-time communication system described further up we perform simulations for several SNRs and different LFSR lengths. The investigated communication system is shown in Figure 2.10 and consist of an LFSR as a transmitter and an NLL as a receiver.

To measure the performance of our receiver we define the two measures *Probability of Synchronization* and the *Time Until Acquisition.*

Definition 2.1 Synchronization *is reached if the state in the receiver is equal to the actual state in the transmitter. The state in the receiver is* *found by a hard-decision on the contents of the delay elements.*

Definition 2.2 *The* Probability of Synchronization $P_{Synch}(k)$ *is the probability that the receiver has reached synchronization after k steps.*

Definition 2.3 *The* Time Until Acquisition (TUA) *is the time k until* $P_{Synch}(k) > \Theta$.

In order to compute the *Probability of Synchronization* we simulate a sequence of 400 bits 10^6 times.

After any time step k we determine how often the receiver has reached *Synchronization* and divide this number by the number of simulations (here 10^6). The *Time Until Acquisition* can be computed based on $P_{Synch}(k)$ as can be seen from its definition.

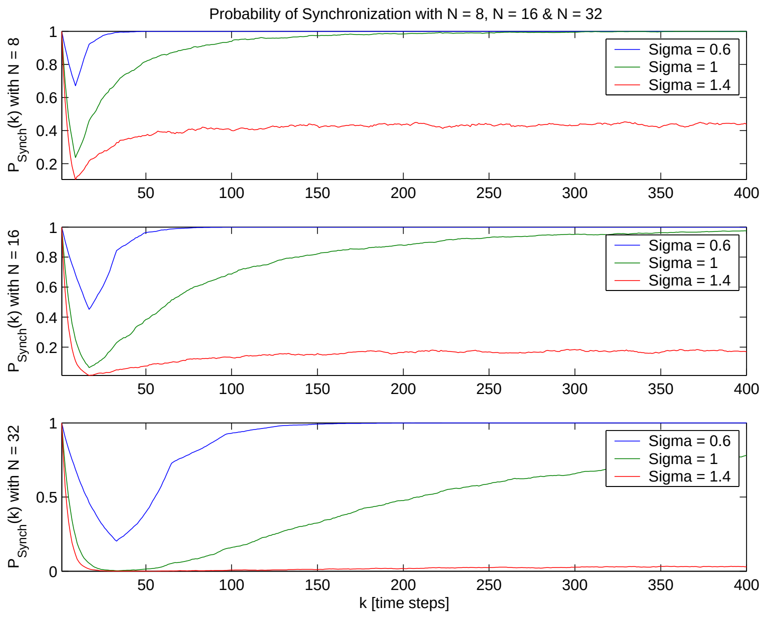

Figure 2.11: Probability of Synchronization after k time steps for different LFSR lengths

These simulations are performed for several LFSR lengths and noise powers. Figure 2.11 shows the *Probability of Synchronization* after k bits. In Figure 2.12 the TUA is shown for $\Theta = 0.9$ for varying SNR values. The SNR is defined as

$$SNR = 10 \log\left(\frac{E_b}{\sigma^2}\right) = 20 \log\left(\frac{1}{\sigma}\right) \tag{2.9}$$

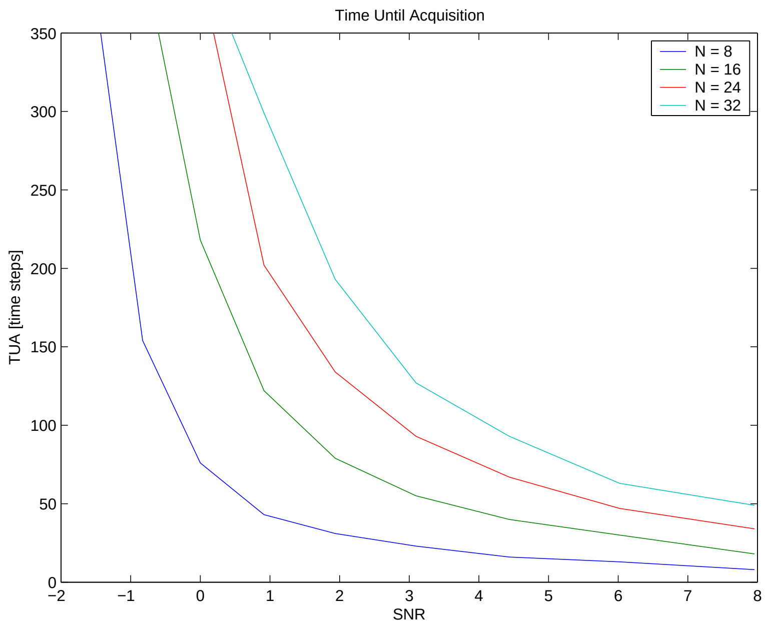

Figure 2.12: Time Until Acquisition for different LFSR lengths

where σ^2 is the variance of the AWGN channel. Equality (2.9) follows because the transmitted signal $\check{x}[k] \in \{+1, -1\}$.

2.4 Discussion

In this Section we discuss the results of the performed simulations shown in Figure 2.11 and 2.12.

Figure 2.11 shows the *Probability of Synchronization* after k bits. From this figure we see that initially $P_{Synch}(k)$ decreases and reaches a minimum at $k = L$, where L is the LFSR length. Then it increases and tends asymptotically to $P_0 \leq 1$. The fact that $P_{Synch}(k)$ decreases for $k < L$ is due to a measurement artefact. In order to determine the state in the receiver a hard-decision is made such that $p\left(x[k]\right) \leq \frac{1}{2}$ is set to 0 and $p\left(x[k]\right) \geq \frac{1}{2}$ is set to 1. Since we initialized the NLL with probabilities $p\left([x[-1] = 1\right) = p\left(x[-2] = 1\right) = \ldots = p\left(x[-L] = 1\right) = \frac{1}{2}$ it is more likely to reach *Synchronization* after the first bits, therefore, the minimum of $P_{Synch}(k)$ is after L bits, when the contents of the register are different to $\frac{1}{2}$.

There are values of σ where $P_0 < 1$. The reason is that in order to achieve *Synchronization* the contents of the delay elements $p\left(x[x-1]\right), p\left(x[k-2]\right), \ldots, p\left(x[k-L]\right)$ need to be close to one or zero and for low SNRs this is not very likely. In contrary a receiver using the Viterbi algorithm on a trellis is able to reach *Synchronization* for every σ. As described in section 2.2 the Viterbi algorithm considers the probabilities of all bits in the past and is therefore able to estimate the state in the transmitter.

Furthermore, we observe that the rise of $P_{Synch}(k)$ depends on the length of the LFSR. This can be explained by the fact that the larger the LFSR length, the more probabilities are stored in the NLL and, therefore, it takes longer to achieve *Synchronization*.

As mentioned in Section 2.2, the NLL considers only the two probabilities $p\left(x[k-M]\right)$ and $p\left(x[k-L]\right)$ and, therefore, it is not possible to improve this receiver by increasing the length.

Figure 2.12 shows the *Time Until Acquisition* (TUA). It can be seen that for low SNRs the TUA becomes very large. Additionally, the larger the LFSR length, the higher the SNR must be to reach the same TUA.

2.5 Outlook

The results of the simulation we performed gave us more insight into the behavior of the Noise-Lock loop. However, some questions remain. For example we do not know yet how to calculate the σ-values above which *Synchronization* cannot be achieved. Furthermore, we have not implemented this communication system in hardware yet.

However, this is a topic of future research. In this semester project we leave the field of discrete-time synchronization at this point and consider the continuous-time case in the next chapter.

Chapter 3

Continuous-Time Synchronization

In Chapter 2 we have presented a discrete-time communication system, consisting of an LFSR as a transmitter and an NLL as a receiver. Both the transmitter and the receiver require a clock signal to trigger the delay cells. Furthermore both clock signals need to be aligned to each other.

In this chapter we present continuous-time communication systems where no clock signals are required.

These systems are discussed as follows: First a pseudo-continuous-time system is investigated. This system is a straightforward extension of the discrete-time system presented in Chapter 2. The discrete-time delay cells are replaced by ideal continuous-time delay cells. Then a continuous-time system is discussed, where the delay cells in both the receiver and the transmitter are replaced by IIR filters. We consider two different approaches to design the IIR filters. In the first approach sequences similar to LFSR sequences are transmitted. In the second approach the transmitted sequences are arbitrary.

For both systems simulations are performed for various noise levels. For the pseudo-continuous-time system we use the *Probability of Synchronization* defined in Section 2.3 as a measure of it performance. For the continuous-time system the *Probability of Synchronization* is not appropriate. Instead, we compute the *Mean-Square-Error* (MSE) between the signals in the transmitter and the signals in the receiver.

We investigate both continuous-time communication systems in the following order. In section 3.1 the pseudo-continuous-time system is explained. In Section 3.2 we discuss the continuous-time system. The results of the simulations are presented in Section 3.3 and discussed in Section 3.4.

3.1 Pseudo-Continuous-Time Synchronization

The pseudo-continuous-time system is a straightforward extension of the discrete-time system. In the following subsection the LFSR in Section 2.1 is being adapted in order to get a continuous-time transmitter. In Subsection 3.1.2 the pseudo-continuous-time NLL is presented.

3.1.1 The Linear Feedback Shift Register

In order to generate continuous-time PN sequences we can use the Linear Feedback Shift Register in Figure 2.3 and transform the discrete-time output into a continuous-time signal [2] as follows

$$x(t) = \sum_k x[k]h(t-kT), \tag{3.1}$$

where

$$h(t) = \begin{cases} 1, & 0 \le t < T \\ 0, & otherwise. \end{cases} \tag{3.2}$$

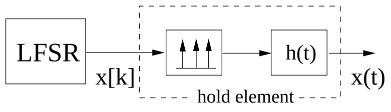

Figure 3.1: Continuous-time LFSR

Figure 3.1 shows a transmitter consisting of a discrete-time LFSR and a hold element. The input of the hold element is the output of the LFSR $x[k]$. The output is a continuous-time signal according to (3.1).

In another approach, the discrete-time delay cells used in the LFSR shown in Section 2.1 could be replaced by ideal continuous-time delay cells. If the outputs of these ideal delay cells are held constant during the time T, then the so designed LFSR generates PN sequences identical to the PN sequences generated with the LFSR shown in Figure 3.1.

However, one cannot perform a computer simulation in continuous-time. Therefore, instead of a continuous-time PN sequence it is sufficient to oversample the discrete-time sequence $x[k]$, which means that N samples per bit are transmitted. We call this oversampled PN sequence a pseudo-continuous-time sequence. Figure 3.2 shows a discrete-time sequence, a pseudo-continuous-time sequence and a continuous-time sequence. The pseudo-continuous-time sequence is the oversampled discrete-time sequence using 5 samples per bit.

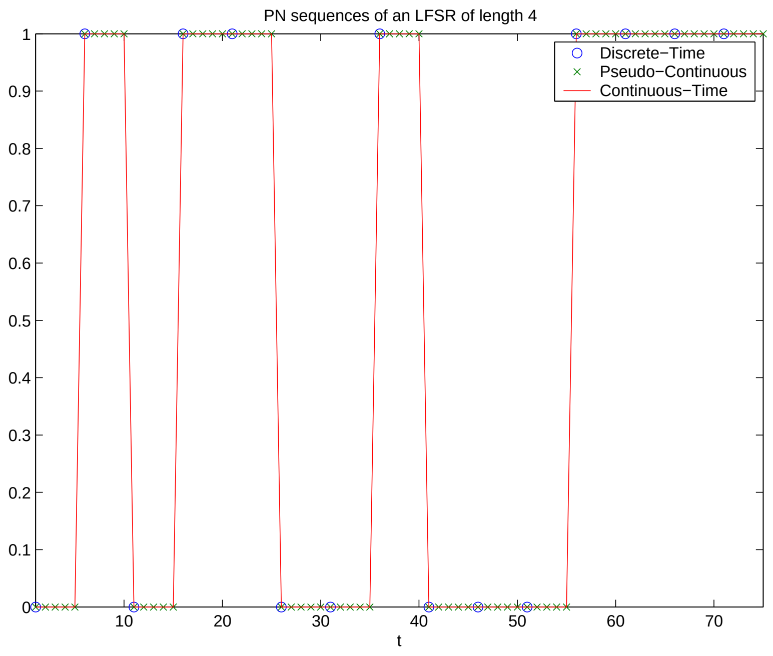

Figure 3.2: PN sequences for discrete-time, pseudo-continuous-time and continuous-time

In the next subsection we present a continuous-time receiver that is deduced from the NLL presented in Section 2.2.

3.1.2 The Noise-Lock Loop

The receiver is similar to the NLL presented in Section 2.2. Again, the receiver works with probabilities instead of bit sequences. The number or delay cells defines the length of the NLL. Figure 3.3 shows an NLL with length 2 with ideal delay cells.

In general, the input of the register $p\left(x(t)\right)$ of an NLL of length L can

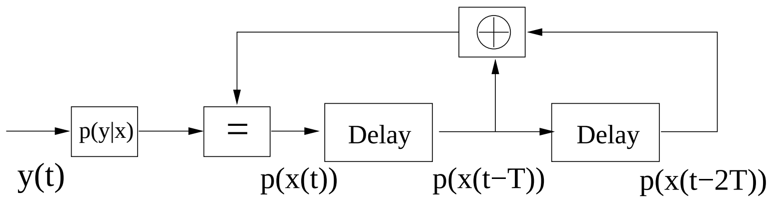

Figure 3.3: Continuous-time Noise-Lock loop

be written as

$$\begin{aligned} p\left(x(t)\right) &= \Big[p\left(x(t-T)\right) \bigoplus p\left(x(t-LT)\right)\Big] \boxed{=} \, p\big(x(t)\,\big|\,y(t)\big) \\ &= p\big(x(t) = (x(t-T) \oplus x(t-LT)) = y(t)\big) \qquad (3.3) \end{aligned}$$

where $\bigoplus$ denotes the Soft-XOR and $\boxed{=}$ the Soft-Equal. This continuous-time receiver computes the probabilities of the signals $x(t), x(t-T), \ldots, x(t-LT)$ given the observation $y(t)$ and is, therefore, the soft-version of an LFSR using ideal delay cells.

In order to design a receiver that is able to synchronize to pseudo-continuous-time sequences that are oversampled with N samples per bit, the ideal delay cells can be replaced by N discrete-time delay cells. We call this receiver a pseudo-continuous-time NLL. As an example, Figure 3.4 shows a pseudo-continuous-time NLL of length two with five discrete-time delay cells instead of one ideal delay cell.

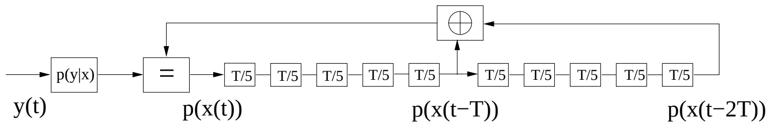

Figure 3.4: Pseudo-continuous-time Noise-Lock loop with NLL length 2 and 5 samples per bit

We have seen that it is possible to design a pseudo-continuous-time communication system by oversampling the discrete-time output of the LFSR with N samples per bit and replacing every ideal delay cell by N discrete-time delay cells. However, the goal of this project is to study a continuous-time system that can be built in hardware. The pseudo-continuous-time approach is not practicable since, in order to emulate the delay cells, storage elements are needed. Furthermore, both the transmitter and the receiver still need a clock signal, however both clock signals do not need to be aligned to each other. pseudo-continuous-time synchronization is therefore only of academic interest. In the next section we present an approach to design a continuous-time communication system that can be realized in hardware.

3.2 Continuous-Time Synchronization

In this section a continuous-time communication system is designed that can be realized in hardware. Therefore, the ideal delay cells are replaced by IIR filters. Same as in Section 3.1 we will first study the continuous-time transmitter. Then in Subsection 3.2.2 the continuous-time receiver is discussed.

3.2.1 Continuous-Time Transmitter

The continuous-time transmitter shown in Figure 3.5 consists of a cascade of two filters, a logic XOR, two Hard Limiters and a mapper. In the following we study these components more detailed.

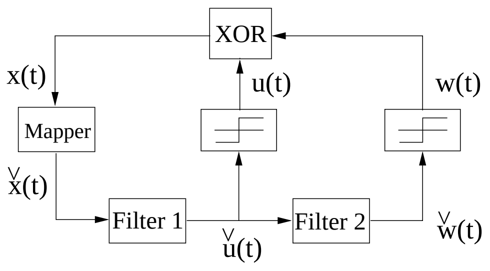

Figure 3.5: Continuous-time transmitter

The mapper maps the logic bits $x(t) \in \{0, 1\}$ to signals $\check{x}(t) \in \{+1, -1\}$ such that

$$\check{x}(t) = 1 - 2x(t). \tag{3.4}$$

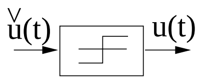

Figure 3.6: Hard Limiter

The Hard Limiter shown in Figure 3.6 transforms the signals $\check{u}(t) \in \mathbb{R}$ back to a bit sequence $u(t) \in \{0, 1\}$ (a negative number is mapped to 1 and a positive to 0, i.e. $\check{u}(t) \leq 0 \rightarrow 1, \check{u}(t) > 0 \rightarrow 0$).

The input $\check{x}(t)$ of *Filter 1* depends on the output of the Hard Limiters $u(t)$ and $w(t)$ and can be written as

$$\check{x}(t) = 1 - 2\big(u(t) \oplus w(t)\big) \tag{3.5}$$

Filter 1 and *Filter 2* are common analog lowpass filters. In order to simulate the continuous case we use the corresponding IIR filters. The choice of the filters affects the characteristics of the generated PN sequence. In the following we present two possible choices of the filters. First, the filters are designed in order to generate signals $\check{u}(t)$ and $\check{w}(t)$ that are similar to the signals in an LFSR using ideal delay cells. Then the filters are chosen arbitrarily, which leads to arbitrary sequences.

LFSR Sequences

We now want to design the filters in order to get signals $\check{u}(t)$ and $\check{w}(t)$ that are similar to the signals in an LFSR using ideal filters. A possible sequence could be

$$\check{u}(t) = \sum_i X_i g(t - iT) \tag{3.6}$$

with $X_i \in \{+1, -1\}$ and $g(t) = 0\ \forall |t| > T/2$. T is the bit length.

$X = \{\ldots, X_{-1}, X_0, X_{+1}, \ldots\}$ is a binary sequence. $\check{u}(t)$ corresponds to the modulation of the sequence on the waveform $g(t)$. Figure 3.7 shows a possible signal generated according to (3.6).

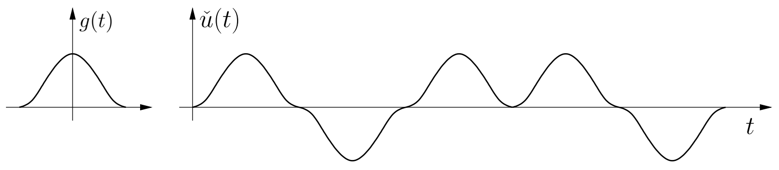

Figure 3.7: The waveform $g(t)$ and a possible sequence of $\check{u}(t)$

However, we were not able to generate a signal where the waveform is independent of the binary sequence X. So we consider the case where the waveform $g(t)$ depends on X_{i-1}, X_i and X_{i+1} (figure 3.8 shows an example) with

$$\check{u}(t) = \sum_i X_i g_{\tilde{X}_{i-1}\tilde{X}_{i+1}}(t - iT) \tag{3.7}$$

where $g_{\tilde{X}_{i-1}\tilde{X}_{i+1}}(t) = 0, |t| > T/2$ and

$$\tilde{X}_{i-1} = \begin{cases} \text{“} = \text{“} & X_{i-1} = X_i \\ \text{“} \neq \text{“} & X_{i-1} \neq X_i \end{cases} \tag{3.8}$$

$$\tilde{X}_{i+1} = \begin{cases} \text{“} = \text{“} & X_{i+1} = X_i \\ \text{“} \neq \text{“} & X_{i+1} \neq X_i. \end{cases} \tag{3.9}$$

This condition can be fulfilled by a cascade of a delay filter and a lowpass filter. The delay filter has a linear phase and therefore a constant group delay in the chosen frequency range. The lowpass filter is needed in order to smoothen the shape of the signal.

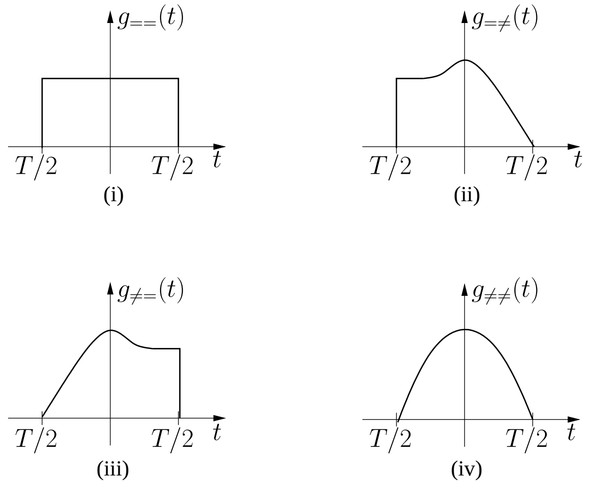

Figure 3.8: Waveforms depending on X_{i-1}, X_i and X_{i+1}. (i) $X_{i-1} = X_i = X_{i+1}$. (ii) $X_{i-1} = X_i \neq X_{i+1}$. (iii) $X_{i-1} \neq X_i = X_{i+1}$. (iv) $X_{i-1} = X_{i+1} \neq X_i$.

Figure 3.9 shows the outputs of *Filter 1* and *Filter 2* in the transmitter in Figure 3.5. Here, *Filter 1* consists of a cascade of a delay filter[1] of order $N = 6$ with a linear phase in the normalized frequency range $\omega_c \in [0, 0.17]$ and a Chebyshev lowpass filter of order $N = 2$ with normalized bandwidth $\omega_c = 0.13$ and peak-to-peak ripple in the passband of $R_p = 0.01$dB. *Filter 2* is the cascade of two of those filters. Figure 3.10 shows a picture of both filters. This transmitter corresponds to an LFSR of length $L = 4$.

As can be seen in Figure 3.9 the signals after *Filter 1* and *Filter 2* consists of two parts; in the first part for $t < 3000$ the signals are aperiodic. Then, after 3000 samples the signals become periodic.

[1]Designed in *Matlab* with the function *iirgrpdelay*.

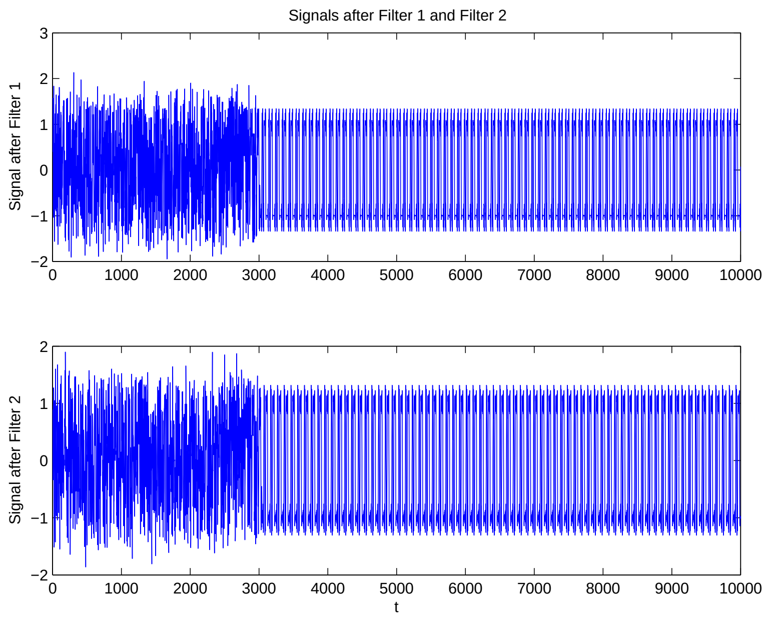

Figure 3.9: Output signals of *Filter 1* and *Filter 2* similar to the signals in an LFSR

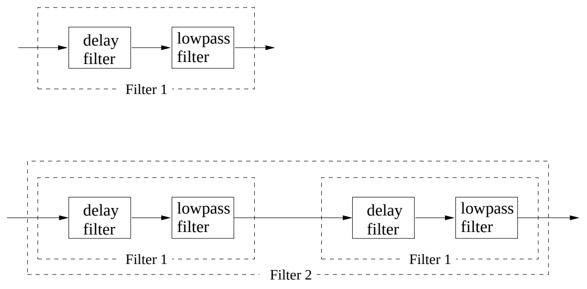

Figure 3.10: *Filter 1* and *Filter 2*

In order to get signals of the form described in (3.7) the outputs of *Filter 1* and *Filter 2* need to be periodic. Therefore, one has to wait 3000 samples until the transmitter generates LFSR sequences.

Figure 3.11 shows a enlarged region of the outputs of *Filter 1* and *Filter 2*.

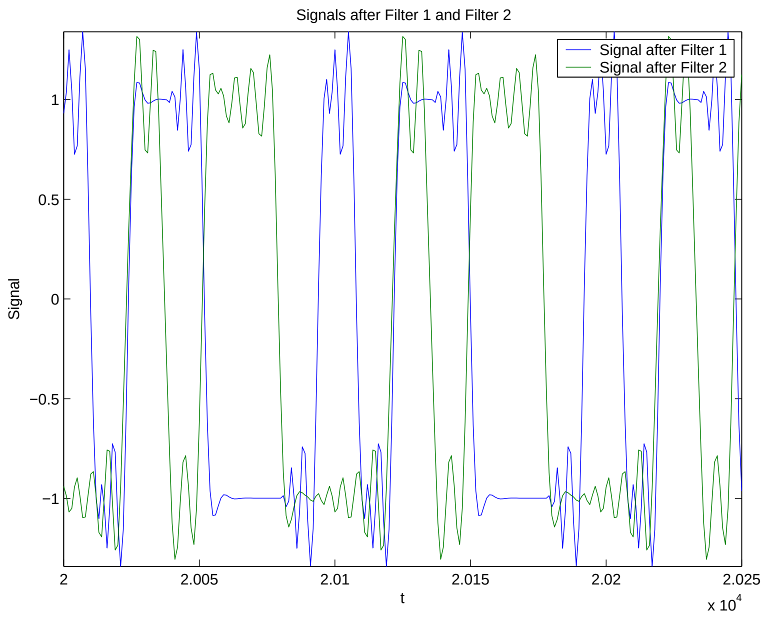

Figure 3.11: Zoom of the output signals of *Filter 1* and *Filter 2* similar to the signals in an LFSR

The PN sequence at the output $\check{x}(t)$ of the continuous-time transmitter shown in Figure 3.12 is identical to the PN sequence generated with the continuous-time LFSR of length $L = 3$ presented in Subsection 3.1.1. Note that the sequence is periodic with a period of $\approx$ 100 samples. The corresponding bit sequence is 1110100. Consequently the spectrum is discrete.

The presented continuous-time transmitter is able to generate PN sequences. However, the larger the delay of *Filter 2*, the more difficult becomes to properly determine the filter parameters. Because of the feedback a small error in the delay distorts the signals so that the assumption of the waveforms that depend only on X_{i-1}, X_i and X_{i+1} is invalid. Therefore, we were only able to design transmitters with short LFSR lengths.

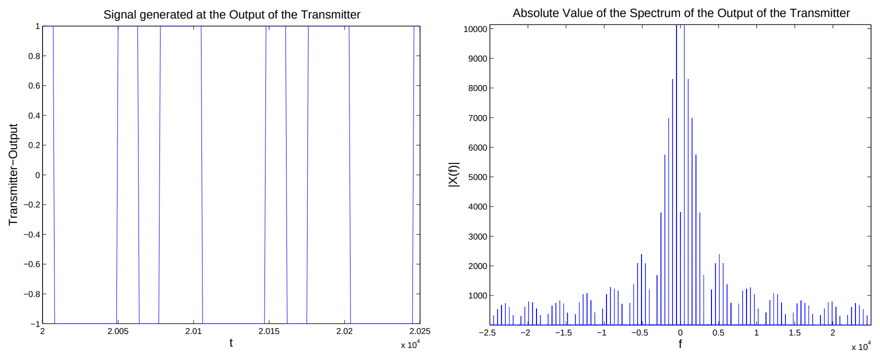

Figure 3.12: LFSR sequence and spectrum at the output of the continuous-time receiver

Arbitrary Sequences

Contrary to the approach above it is now not the goal of this approach to have LFSR sequences according to (3.7). Instead, the filters are chosen arbitrary.

Figure 3.13 shows the output signals $\breve{u}(t)$ and $\breve{w}(t)$ of both filters in the transmitter. *Filter 1* is a Chebyshev lowpass filter of order 5 with normalized frequency $\omega_c = 0.02$ and peak-to-peak ripple in the passband of $R_p = 0.01$dB. *Filter 2* is the cascade of three of those filters.

The PN sequence at the output $\breve{x}(t)$ of the continuous-time transmitter with arbitrarily chosen filters is shown in Figure 3.14. Regarding the spectrum of the PN sequence it can be seen that the generated sequence is either aperiodic or has a very large period.

As shown in Figure 3.14 the continuous-time transmitter with arbitrarily chosen filters is able to generate PN sequences that have a continuous spectrum. As we have chosen the parameters of the filters arbitrarily, it is less demanding to design the transmitter, even if *Filter 2* has a large delay.

3.2.2 Continuous-Time Receiver

The continuous-time receiver shown in Figure 3.15 consists of two filters, a Soft-Equal and a Soft-XOR gate, the block $p(y|x)$, a mapper and the *Soft Limiter*. Again this receiver runs with probabilities instead of bit sequences.

The Soft-Equal and the Soft-XOR gate are equal to the soft gates described in Section 2.2. The Soft-Equal gate computes the probability $p(z)$ given $p(x)$ and $p(y)$ and the condition $z = x = y$. The Soft-XOR gate com-

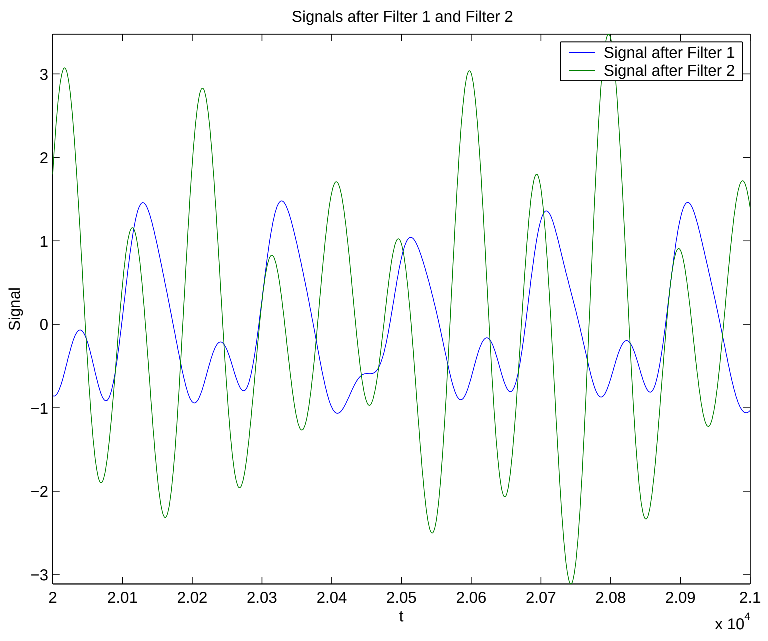

Figure 3.13: Output signals of *Filter 1* and *Filter 2* for arbitrarily chosen filters

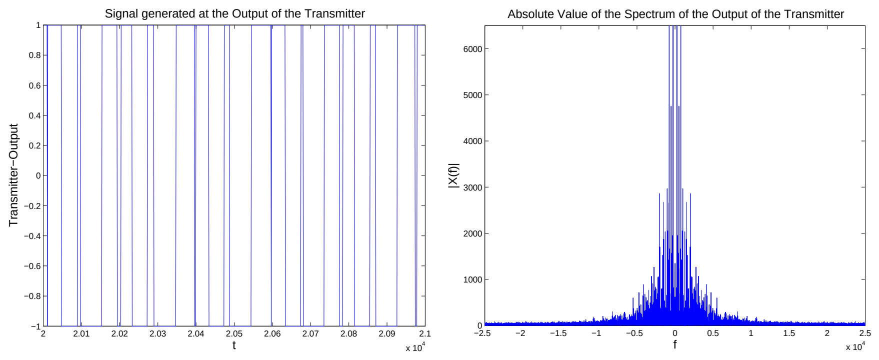

Figure 3.14: Signal and spectrum at the output of the continuous-time transmitter for arbitrary chosen filters

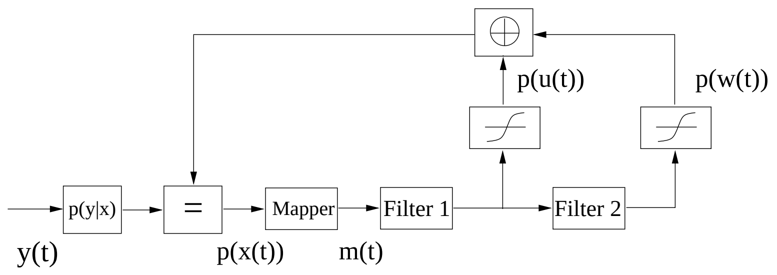

Figure 3.15: Continuous-Time Receiver

putes the probability $p(z)$ given $p(x)$ and $p(y)$ and the condition $z = x \oplus y$. The block $p(y|x)$ has the same function as in the NLL since the channel is still modelled as an AWGN channel.

The mapper computes the expectation of $\check{x}(t)$ given the observation $y(t)$:

$$\begin{aligned} m(t) &= \mathrm{E}\left[\check{x}(t)\right] \\ &= p\left(\check{x}(t) = 1\right) - p\left(\check{x}(t) = -1\right) \\ &= p\left(x(t) = 0\right) - p\left(x(t) = 1\right) \end{aligned} \tag{3.10}$$

where the second equality follows from (3.4).

Filter 1 and *Filter 2* are identical to the filters in the continuous-time transmitter presented in Subsection 3.2.1. $m_{Filter1}$ and $m_{Filter2}$ denote the outputs of *Filter 1* resp. *Filter 2*. In the following we proof that $m_{Filter1}$ and $m_{Filter2}$ are the expectations $\mathrm{E}\left[\check{u}(t)\right]$ and $\mathrm{E}\left[\check{w}(t)\right]$ of the outputs $\check{u}(t)$ and $\check{w}(t)$ of the filters in the transmitter shown in Figure 3.5:

$$\check{u}(t) = \left(\check{x}(t) * h_{Filter1}\right)(t) \tag{3.11}$$

$$\check{w}(t) = \left(\check{u}(t) * h_{Filter2}\right)(t) \tag{3.12}$$

and

$$\begin{aligned} \mathrm{E}\left[\check{u}(t)\right] &= \mathrm{E}\left[\int h_{Filter1}(\tau)\check{x}(t-\tau)\,\mathrm{d}\tau\right] \\ &= \int h_{Filter1}(\tau)\mathrm{E}\left[\check{x}(t-\tau)\right]\,\mathrm{d}\tau \\ &= \left(h_{Filter1} * m\right)(t) \\ &= m_{Filter1}(t) \end{aligned} \tag{3.13}$$

where the last equality follows from (3.10).

Likewise

$$\begin{aligned} \mathrm{E}\left[\check{w}(t)\right] &= \left(h_{Filter2} * m_{Filter1}\right)(t) \\ &= m_{Filter2}(t) \end{aligned} \tag{3.14}$$

where the last equality follows from (3.13). *Filter 1* and *Filter 2* in the continuous-time receiver filter the expectations of the corresponding signals in the continuous-time transmitter.

Figure 3.16 shows a Soft Limiter. The Soft Limiter computes the probability of the signal $u(t)$ given the expectation $m(t)$ at the input. In order

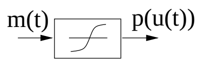

Figure 3.16: Soft Limiter

to design the Soft Limiter, knowledge of the probability density functions of the filter outputs $m_{Filter1}$ and $m_{Filter2}$ is needed. Consequently, the Soft Limiter depends on the choice of the filters.

First, the approach leading to LFSR sequences is investigated. Then, the approach, where the filters are chosen arbitrarily, is studied.

LFSR Sequences

As presented in Subsection 3.2.1 the signals $\check{u}(t)$ and $\check{w}(t)$ can be written as

$$\check{u}(t) = \sum_i X_i g_{\tilde{X}_{i-1}\tilde{X}_{i+1}}(t - iT - \tau) \tag{3.15}$$

and likewise

$$\check{w}(t) = \sum_j X_j q_{\tilde{X}_{j-1}\tilde{X}_{j+1}}(t - jT - \tau) \tag{3.16}$$

with $X_i, X_j \in \{1, -1\}$ and where the waveforms $g_{\tilde{X}_{i-1}\tilde{X}_{i+1}}(t) = 0\ \forall |t| > T/2$ and $q_{\tilde{X}_{j-1}\tilde{j+1}}(t) = 0\ \forall |t| > T/2$. τ is a stochastic variable and denotes the unknown time shift of the signal. Note that the waveforms depend on X_{i-1}, X_i and X_{i+1} resp. X_{j-1}, X_j and X_{j+1}.

The Soft Limiter computes the probabilities of $p\,(u(t))$ and $p\,(w(t))$ given the inputs $m_{Filter1}$ and $m_{Filter2}$. However, it is sufficient to compute the probabilities $p\,(X_i)$ and $p\,(X_j)$. The reason is, that the outputs of the Hard Limiters $u(t)$ and $w(t)$ can be written as

$$\begin{aligned} u(t) &= \frac{1}{2}\Big(1 - \operatorname{sign}(\check{u}(t))\Big) \\ &= \frac{1}{2}(1 - X_i) \end{aligned} \tag{3.17}$$

and similarly

$$w(t) = \frac{1}{2}(1 - X_j) \tag{3.18}$$

where sign(x) denotes the signum-function. The equations (3.17) and (3.18) follow from (3.15), (3.16) and the fact that the waveforms $g_{\tilde{X}_{i-1}\tilde{X}_{i+1}}(t) \geq 0$ and $q_{\tilde{X}_{j-1}\tilde{j+1}}(t) \geq 0$.

The waveforms depend on X_{i-1}, X_i, X_{i+1} resp. X_{j-1}, X_j, X_{j+1}. We assume the bits X_i and X_j to be unknown and thus:

$$p\left(X_{i-1}=0\right)=p\left(X_{i-1}=1\right)=p\left(X_{i+1}=0\right)=p\left(X_{i+1}=1\right)=\frac{1}{2} \tag{3.19}$$

$$p\left(X_{j-1}=0\right)=p\left(X_{j-1}=1\right)=p\left(X_{j+1}=0\right)=p\left(X_{j+1}=1\right)=\frac{1}{2}. \tag{3.20}$$

Theorem 3.1 *Assuming the probabilities $p\left(X_i\right)$ and $p\left(X_j\right)$ to be $\frac{1}{2}$, the outputs of the Soft Limiters $p\left(X_i=1\right)$ and $p\left(X_j=1\right)$ can be written as*

$$p\left(X_i=1\right) = \frac{1}{2}\left(1+\frac{m_{Filter1}(t)}{\kappa_1(m_{Filter1})}\right) \tag{3.21}$$

$$p\left(X_j=1\right) = \frac{1}{2}\left(1+\frac{m_{Filter2}(t)}{\kappa_2(m_{Filter2})}\right) \tag{3.22}$$

where

$$\kappa_1(m) = \frac{\mathrm{E}_m\left[g_{==}(\theta)\right]+\mathrm{E}_m\left[g_{=\neq}(\theta)\right]+\mathrm{E}_m\left[g_{\neq=}(\theta)\right]+\mathrm{E}_m\left[g_{\neq\neq}(\theta)\right]}{4} \tag{3.23}$$

$$\kappa_2(m) = \frac{\mathrm{E}_m\left[q_{==}(\theta)\right]+\mathrm{E}_m\left[q_{=\neq}(\theta)\right]+\mathrm{E}_m\left[q_{\neq=}(\theta)\right]+\mathrm{E}_m\left[q_{\neq\neq}(\theta)\right]}{4} \tag{3.24}$$

with

$$\theta = t-\tau \tag{3.25}$$

and

$$\mathrm{E}_m\left[g(\theta)\right] = \int_{D_1(m)} g(\theta)f(\theta)\,\mathrm{d}\theta \tag{3.26}$$

$$D_1(m) = \left\{\theta\in[-T/2,T/2] \;\middle|\; g(\theta)\geq |m|\right\} \tag{3.27}$$

resp.

$$\mathrm{E}_m\left[q(\theta)\right] = \int_{D_2(m)} q(\theta)f(\theta)\,\mathrm{d}\theta \tag{3.28}$$

$$D_2(m) = \left\{\theta\in[-T/2,T/2] \;\middle|\; q(\theta)\geq |m|\right\} \tag{3.29}$$

where $f(\theta)$ denotes the probability density function of θ.

Proof: See appendix A.

The algorithm used to compute $\mathrm{E}_m\left[g(\theta)\right]$ and $\mathrm{E}_m\left[q(\theta)\right]$ is shown in Figure 3.17. It works as follows: θ is chosen from a uniform distribution. If $g(\theta)\in D(m)$ (i.e. $g(\theta)\geq|m|$), then $g(\theta)$ is used to compute $\mathrm{E}_m\left[g(\theta)\right]$

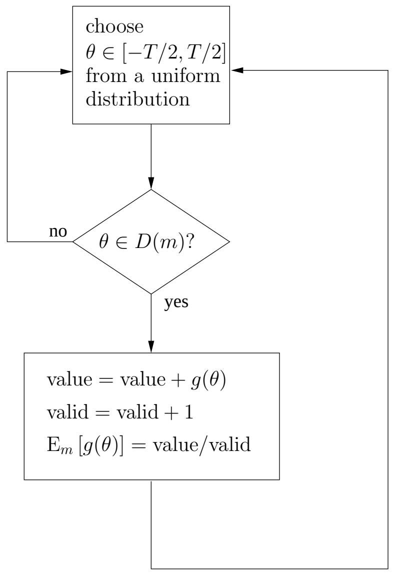

Figure 3.17: Algorithm to compute $\mathrm{E}_m[g(\theta)]$

resp. $\mathrm{E}_m[q(\theta)]$. If θ is not valid, another one is chosen. This is done for several iterations.

The algorithm is executed for different expectations m. By substituting $\mathrm{E}_m[g(\theta)]$ and $\mathrm{E}_m[q(\theta)]$ in (3.26) and (3.28) by the expectations computed with this algorithm one can determine the probabilities $p(X_i = 1)$ and $(X_j = 1)$. Figure 3.18 shows the so computed probabilities $p(X_i = 1)$ and $p(X_j = 1)$ of a continuous-time transmitter which generates LFSR sequences.

In the following we investigate the continuous-time receiver where arbitrary sequences are generated.

Arbitrary Sequences

In contrast to the approach that leads to LFSR sequences, the filter outputs here are not periodic. Consequently, a lot more waveforms would be necessary to describe those signals. Therefore, we propose a waveform by

$$\check{u}(t) = (1 - 2u(t))\, g(t - \tau) \tag{3.30}$$

and likewise

$$\check{w}(t) = (1 - 2w(t))\, q(t - \tau) \tag{3.31}$$

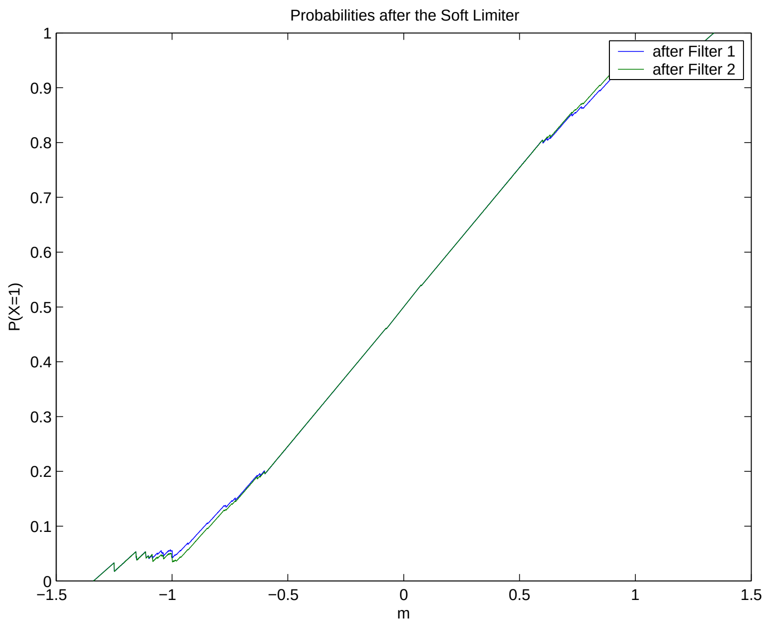

Figure 3.18: $p\left(X_i = 1\right)$ and $p\left(X_j = 1\right)$

where $u(t)$ and $w(t)$ are the outputs of the Hard Limiter in the transmitter. $\breve{u}(t)$ is the modulation of the bit $u(t)$ on the waveform $g(t)$. The same holds for $\breve{w}(t)$. The waveforms $g(t) \geq 0$ and $q(t) \geq 0$ are stochastic. τ is the unknown time shift of the signals.

The expectations $m_{Filter1}(t)$ and $m_{Filter2}(t)$ can be written as

$$\begin{aligned} m_{Filter1}(t) &= \mathrm{E}_{m_{Filter1}}\left[\breve{u}(t)\right] \\ &= \mathrm{E}_{m_{Filter1}}\left[\left(1-2u(t)\right)g(t-\tau)\right] \\ &= \mathrm{E}\left[1-2u(t)\right]\mathrm{E}_{m_{Filter1}}\left[g(\theta)\right] \end{aligned} \tag{3.32}$$

and similarly

$$m_{Filter2}(t) = \mathrm{E}\left[1-2w(t)\right]\mathrm{E}_{m_{Filter2}}\left[q(\theta)\right] \tag{3.33}$$

with $\theta = t - \tau$. The equalities (3.32) and (3.33) follow from the fact that the waveforms do not depend on the bits $u(t)$ and $q(t)$. $\mathrm{E}_m\left[g(\theta)\right]$ and $\mathrm{E}_m\left[q(\theta)\right]$ denote

$$\mathrm{E}_m\left[g(\theta)\right] = \int_{D_1(m)} g(\theta) f(\theta)\,\mathrm{d}\theta \tag{3.34}$$

$$D_1(m) = \left\{\theta \in [-T/2, T/2] \;\middle|\; g(\theta) \geq |m|\right\} \tag{3.35}$$

resp.

$$\begin{aligned} \mathrm{E}_m\left[q(\theta)\right] &= \int_{D_2(m)} q(\theta) f(\theta)\,\mathrm{d}\theta && (3.36) \\ D_2(m) &= \left\{\theta \in [-T/2, T/2] \,\middle|\, q(\theta) \geq |m|\right\} && (3.37) \end{aligned}$$

where $f(\theta)$ is the probability density function of θ.

In the following we assume the time shift τ resp. θ to be uniformly distributed. Note that $|m|$ cannot be greater than $g(\theta)$ or $q(\theta)$ because

$$\begin{aligned} |m(t)| &= \left|\mathrm{E}\left[1-2u(t)\right]\mathrm{E}_m\left[g(\theta)\right]\right| && (3.38) \\ &= \left|\mathrm{E}\left[1-2u(t)\right]\right|\left|\mathrm{E}_m\left[g(\theta)\right]\right| && (3.39) \\ &\leq \left|1-2u(t)\right|\mathrm{E}_m\left[g(\theta)\right] && (3.40) \\ &= \mathrm{E}_m\left[g(\theta)\right] && (3.41) \end{aligned}$$

and likewise

$$\begin{aligned} |m(t)| &\leq \left|1-2w(t)\right|\mathrm{E}_m\left[q(\theta)\right] && (3.42) \\ &= \mathrm{E}_m\left[q(\theta)\right] && (3.43) \end{aligned}$$

where (3.40) and (3.42) follow from the fact that $u(t), w(t) \in \{0,1\}$ and $g(\theta), q(\theta) \geq 0$.

In order to compute the probabilities $p\left(u(t)=0\right)$ and $p\left(w(t)=0\right)$, it suffices to know the expectations $\mathrm{E}\left[1-2u(t)\right]$ and $\mathrm{E}\left[1-2w(t)\right]$ since

$$\begin{aligned} p\left(u(t)=0\right) &= p\left((1-2u(t))=1\right) \\ &= \frac{1}{2}\left(1+\mathrm{E}\left[1-2u(t)\right]\right) \\ &= \frac{1}{2}\left(1+\frac{m_{Filter1}(t)}{\mathrm{E}_{m_{Filter1}(t)}\left[g(\theta)\right]}\right) && (3.44) \end{aligned}$$

and similarly

$$p\left(w(t)=0\right) = \frac{1}{2}\left(1+\frac{m_{Filter2}(t)}{\mathrm{E}_{m_{Filter2}}\left[q(\theta)\right]}\right). \qquad (3.45)$$

The equalities (3.44) and (3.45) follow from (3.32) and (3.33).

In order to compute $\mathrm{E}_{m_{Filter1}}\left[g(t)\right]$ and $\mathrm{E}_{m_{Filter2}}\left[q(t)\right]$ we use the algorithm shown in Figure 3.17. The stochastic waveforms $g(t)$ and $q(t)$ are approximated by the absolute values of the filter outputs $\check{u}(t)$ and $\check{w}(t)$.

Figure 3.19 shows the probabilities $p\left(u(t)=0\right)$ and $p\left(w(t)=0\right)$ computed with this algorithm shown in Figure 3.17, compared to the probabilities computed according to (3.46), (3.47), (3.48) and (3.49).

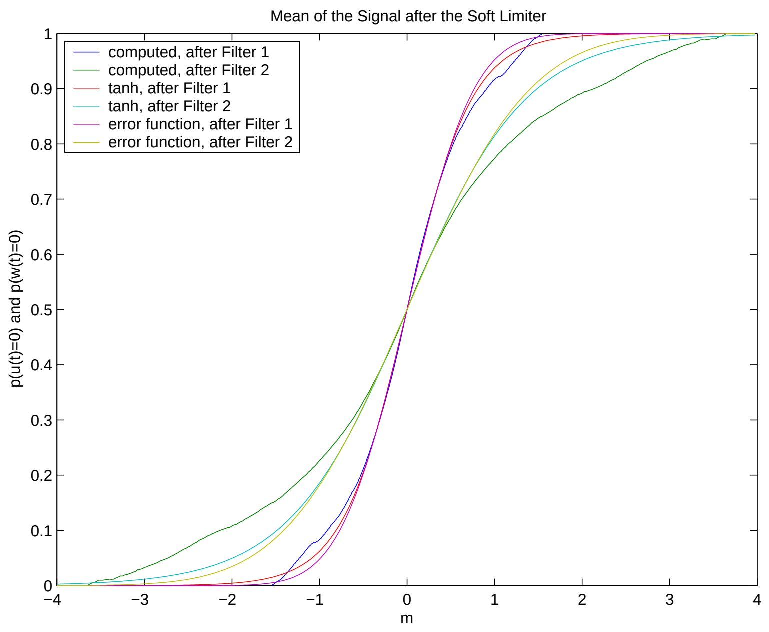

Figure 3.19: Probabilities $p\,(u(t) = 0)$ and $p\,(w(t) = 0)$ computed with the algorithm, compared to the probabilities computed according to (3.46),(3.47),(3.48) and (3.49)

The probabilities computed by the algorithm can be well approximated by the following probabilities:

$$p\left(u(t)=0\right) = \frac{1}{2}\left(1+\operatorname{erf}\left(\frac{m_{Filter1}(t)}{\sqrt{2}\varsigma_1}\right)\right) \quad (3.46)$$

$$P\left(w(t)=0\right) = \frac{1}{2}\left(1+\operatorname{erf}\left(\frac{m_{Filter2}(t)}{\sqrt{2}\varsigma_2}\right)\right) \quad (3.47)$$

or

$$p\left(u(t)=0\right) = \frac{1}{2}\left(1+\tanh\left(\frac{1.15 m_{Filter1}(t)}{\sqrt{2}\varsigma_1}\right)\right) \quad (3.48)$$

$$P\left(w(t)=0\right) = \frac{1}{2}\left(1+\tanh\left(\frac{1.15 m_{Filter2}(t)}{\sqrt{2}\varsigma_2}\right)\right) \quad (3.49)$$

with $\varsigma_1 = 0.6$ and $\varsigma_2 = 1.1$. Note that the Hard Limiter maps a negative value at the input to a 1 and a positive to a 0. Therefore, the probabilities can be written as

$$p\left(u(t)=0\right) = p\left(\breve{u}(t) > 0\right) \quad (3.50)$$

$$p\left(w(t)=0\right) = p\left(\breve{w}(t) > 0\right). \quad (3.51)$$

If we consider the *Central Limit Theorem* [4] and assume $\breve{u}(t)$ to be Gaussian distributed with mean $m_{Filter1}$ and variance ς_1^2, equation (3.50) leads to (3.46). Similarly, if we assume $\breve{w}(t)$ to be Gaussian distributed with mean $m_{Filter2}$ and variance ς_2^2, the equation (3.51) leads to (3.47). Therefore, the equations (3.46) and (3.47) are the probabilities of a Gaussian variable being greater than zero [1].

In (3.48) and (3.49) we approximate the error function with $\tanh(1.15x)$.

In order to built a continuous-time receiver in hardware, it is necessary to use components that can be easily implemented. The implementation of the IIR filters can be done without any problems. In addition, the Soft-Equal and the Soft-XOR gates have already been designed in hardware. Furthermore, it is known how to implement the block $p(y|x)$ [5].

The implementation of the Hard Limiter using the probabilities computed with the algorithm seems to be difficult. Therefore, especially the approach, where the probabilities are computed according to (3.48) and (3.49) is very interesting, since the $\tanh(x)$ can be emulated easily with a circuit consisting of two transistors as shown in Figure 3.20 [6].

We have presented a continuous-time system that can be implemented in hardware. In the next section we show results of the simulations we performed in order to investigate this system.

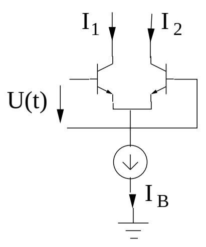

Figure 3.20: Circuit with $I_1 - I_2 = I_B tanh\left(\frac{U(t)}{2U_T}\right)$

3.3 Results

In Sections 3.1 and 3.2 two possible continuous-time communication systems are presented. In the following we show results of simulations of both systems.

The results of the pseudo-continuous-time communication system are investigated in Subsection 3.3.1. Then in Subsection 3.3.2 we present the results of the continuous-time system.

3.3.1 Results of the Pseudo-Continuous-Time System

The pseudo-continuous-time and the discrete-time communication system are very similar. As a matter of fact, the discrete-time system can be interpreted as a special case of a pseudo-continuous-time one, where the output of the LFSR is oversampled with one sample per bit. As a consequence, the simulations and the results for pseudo-continuous-time are very similar to the simulations of the discrete-time system. Again we use the *Probability of Synchronization* ($P_{Synch}(t)$) defined in Chapter 2 as a measure of the performance.

In order to compute $P_{Synch}(t)$ we generate a PN sequence with length of 400 with an LFSR and oversample the output of the discrete-time LFSR with 10 samples per bit. These simulations are performed for several LFSR lengths and noise powers. Figure 3.21 shows $P_{Synch}(t)$.

3.3.2 Results of the Continuous-Time System

We investigate the continuous-time system shown in Figure 3.22. In Subsection 3.3.1 we use the *Probability of Synchronization* as a measure of the performance of the pseudo-continuous-time system. However, this quan-

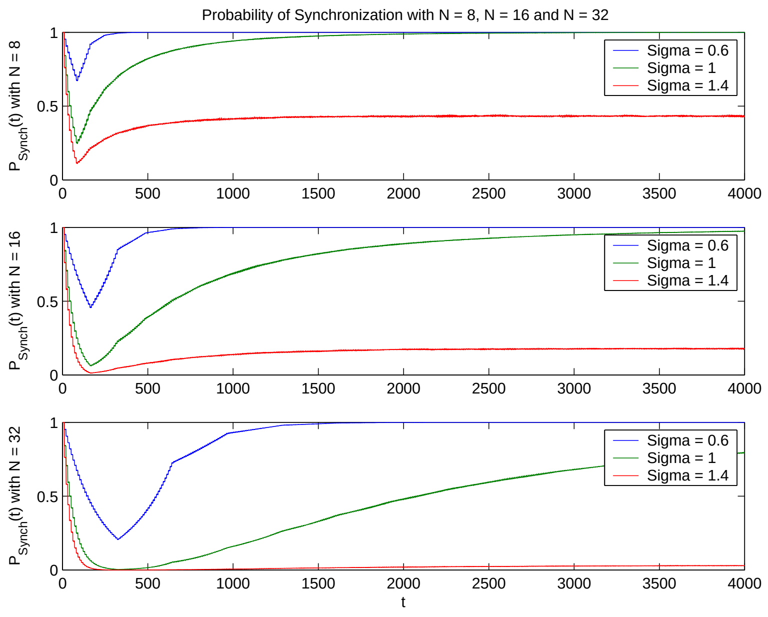

Figure 3.21: The *Probability of Synchronization* of the pseudo-continuous-time system after time t for different LFSR lengths

tity is not appropriate for the continuous-time system. In contrast to the pseudo-continuous-time transmitter, where the outputs of the delay cells are constant over one bit, the outputs of the filters vary in the continuouos-time transmitter. A small signal is more sensitive to noise than a large one, therefore, $P_{Synch}(t)$ depends on the values of the filter outputs. Consequently, $P_{Synch}(t)$ varies over one bit and is not useful as a measure of the system performance.

We measure the performance of the continuous-time receiver by computing the *Mean-Square-Error* (MSE) between the signals $\check{u}(t)$ and $\check{w}(t)$ in the transmitter and the signals $p\left(\check{u}(t)\right)$ and $p\left(\check{w}(t)\right)$ in the receiver. The MSE is the average of the *Square-Error* (SE) over several realizations of the simulations. The SE is defined as

$$SE(t) = \frac{SE_1(t) + SE_2(t)}{2} \tag{3.52}$$

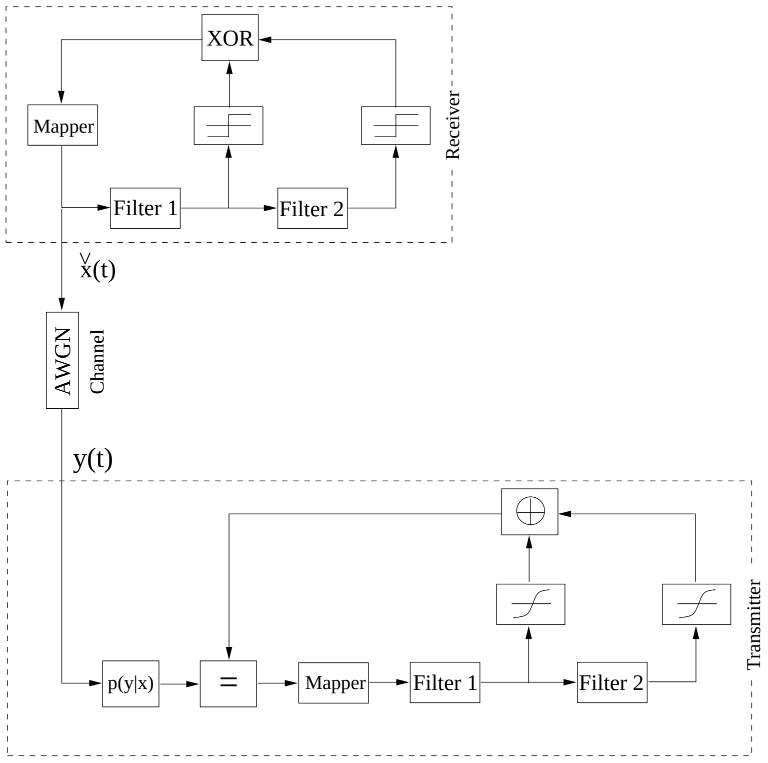

Figure 3.22: Continuous-time communication system

with

$$SE_1(t) = \frac{\left[\check{u}(t) - m_{Filter1}(t)\right]^2}{4 \max\left\{\check{u}^2(t), m^2_{Filter1}(t)\right\}} \tag{3.53}$$

$$SE_2(t) = \frac{\left[\check{w}(t) - m_{Filter2}(t)\right]^2}{4 \max\left\{\check{w}^2(t), m^2_{Filter2}(t)\right\}}. \tag{3.54}$$

where $SE_1(t)$ and $SE_2(t)$ are normalized to one.

In order to measure the performance of the continuous-time receiver we generate a sequence of $50'000$ samples using the continuous-time transmitter presented in Subsection 3.2.1. After any time t we compute the SE and average over 5000 realisations. This simulation is performed for both continuous-time communication systems for several noise powers.

First the results of the LFSR sequences transmitting system are presented. Figure 3.23 shows the signals $\check{u}(t)$ and $m_{Filter1}(t)$ resp. $\check{w}(t)$ and $m_{Filter2}(t)$ evaluated in a simulation with noise power $\sigma^2 = 0.6^2$. In Figure 3.24 the MSE is shown, computed for different noise powers.

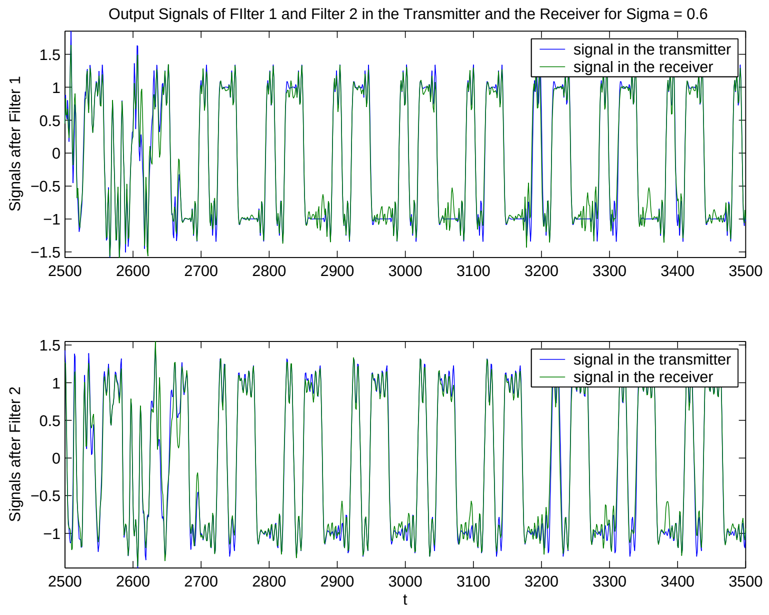

Figure 3.23: Signals $u\check{(}t), w\check{(}t), m_{Filter1}$ and $m_{Filter2}$ in the LFSR sequences transmitting system

In the following the results of the arbitrary sequences transmitting system are presented. In Figure 3.25 the signals $\check{u}(t)$ and $m_{Filter1}$ resp. $\check{w}(t)$ and

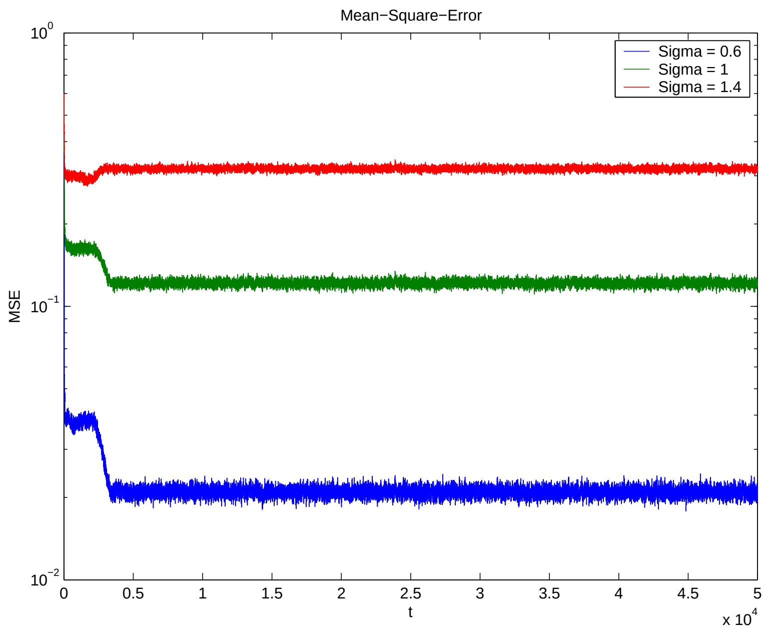

Figure 3.24: The MSE of the LFSR sequences transmitting system

$m_{Filter2}$ are shown, evaluated in a simulation with noise power $\sigma^2 = 0.6^2$. The receiver determines the probabilities $p\,(u(t) = 0)$ and $p\,(w(t) = 0)$ with the numerically computed values using the algorithm shown in Figure 3.17.

Figure 3.26 shows the MSE. The probabilities $p\,(u(t) = 0)$ and $p\,(w(t) = 0)$ are computed in three different ways: First the probabilities are determined using the numerically computed values of $\mathrm{E}\,[g(\theta)]$ and $\mathrm{E}\,[q(\theta)]$. The second and the third probabilities are computed according to (3.48) and (3.49). In this case $\varsigma_1 = 0.6$ and $\varsigma_2 = 1.1$.

3.4 Discussion

Section 3.3 shows the results of the simulations performed for the pseudo-continuous-time and the continuous-time communication system. In the following two subsections we discuss those results. In Subsection 3.4.1 we investigate the results of the pseudo-continuous-time system. In Subsection 3.4.2, we study the results of the continuous-time system.

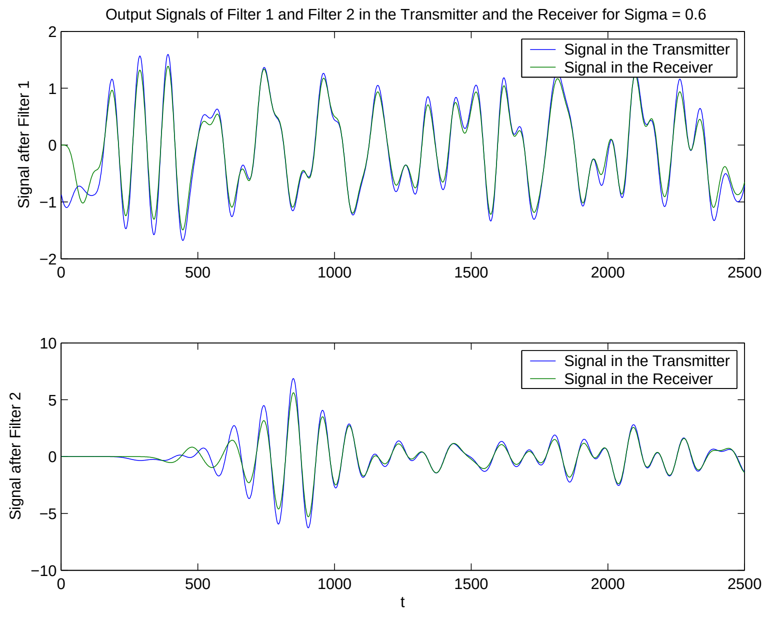

Figure 3.25: Signals $\check{u}(t), \check{w}(t), m_{Filter1}$ and $m_{Filter2}$ in the arbitrary sequences transmitting system

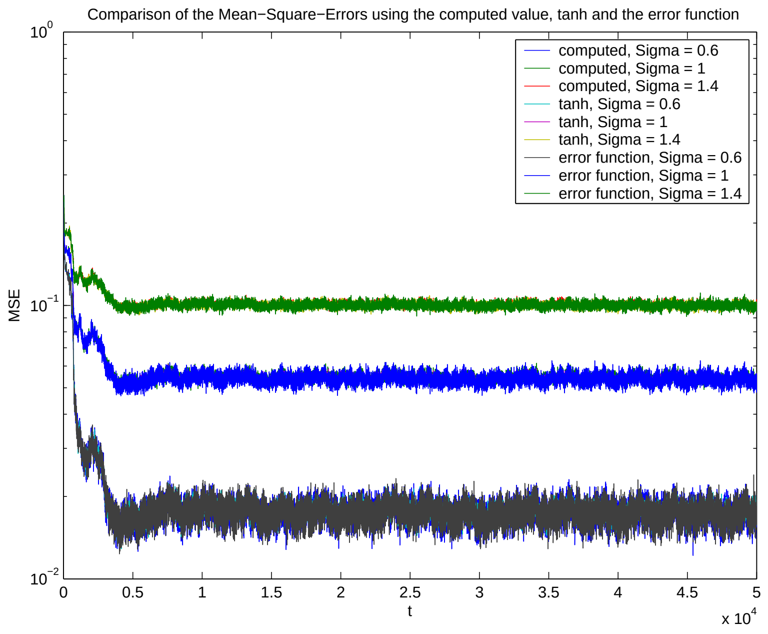

Figure 3.26: The MSE of the arbitrary sequences transmitting system

3.4.1 Discussion of the Results of the Pseudo-Continuous-Time System

In Figure 3.21 the *Probability of Synchronization* after time t is shown. The discrete-time output of the LFSR is oversampled with 10 samples per bit. We can see that $P_{Synch}(t)$ is equal to $P_{Synch}(10k)$.

In order to explain this result we consider the pseudo-continuous-time NLL shown in Figure 3.27 with LFSR length two and two samples per bit.

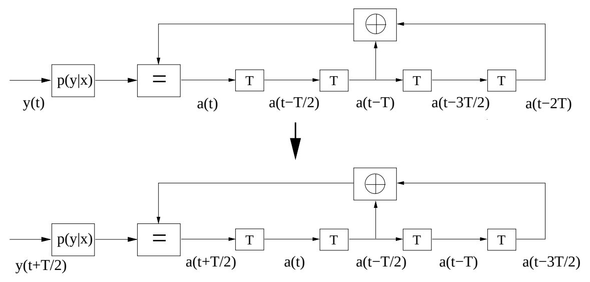

Figure 3.27: Pseudo-continuous-time Noise-Lock loop with LFSR length 2 and 2 samples per bit

We denote the signals after the delay cells with $a(t)$. We see that $a(t)$ only depends on $a(t-T), a(t-2T)$ and $y(t)$. Likewise depends $a(t+T/2)$ only on $a(t-T/2), a(t-3T/2)$ and $y(t+T/2)$. Therefore, this pseudo-continuous-time NLL can be divided into a cascade of two discrete-time NLLs. Note that both NLLs are independent. Consequently, $P_{Synch}(t) = P_{Synch}(2k)$ and in general $P_{Synch}(t) = P_{Synch}(Nk)$.

The simulation of the pseudo-continuous-time communication system leads to the same results obtained in the discrete-time simulation. In the next subsection we discuss the results of the simulations performed with the continuous-time system.

3.4.2 Discussion of Results of the Continuous-Time System

Figure 3.23 and Figure 3.25 show the signals $\check{u}(t), \check{w}(t), m_{Filter1}$ and $m_{Filter2}$ in both communication systems.

We can see that for the chosen value $\sigma = 0.6$ both receivers are able to synchronize to the generated sequences.

Figure 3.24 and Figure 3.26 show the MSE computed for both systems. By comparing both figures we observe, that the continuous-time system that transmits LFSR sequences has a higher MSE than the arbitrary sequences transmitting system. The reason is, that in the second system the filters have a lower bandwidth and, therefore, more noise is filtered out. Consequently, especially with large noise powers this communication system has a better performance.

3.5 Outlook

Several possibilities to design a continuous-time communication system were presented in this chapter. Furthermore, we have shown that the presented receivers are able to synchronize to the generated *PN sequences*.

However, Continuous-Time Synchronization is just in its infancy and a lot of questions remain. For example, the choice of the filters needs to be investigated in more detail. Furthermore, the designed *Soft Limiter* may be improved by assuming another probability density function for the time shift τ. Additionally, it will be necessary to investigate the performance of the continuous-time receiver and compare it to the discrete-time receiver. Moreover it is not clear yet, how to use the proposed communication system to transmit information. And, last but not least, this continuous-time system should be implemented in hardware.

Chapter 4

Conclusion

In this semester project we designed novel low-complexity PN sequence synchronizers, referred to as Noise Lock Loops (NLL).

An NLL is interesting for two reasons. In classical spread spectrum communication systems PN synchronization is separated into two phases: There is an initial acquisition phase and a tracking phase after the signal has been acquired. Acquisition and tracking are typically performed by two separate synchronization systems. The NLL consists of one single system of low-complexity that both acquires and tracks noisy PN sequences. Furthermore, classical PN synchronizers are clocked. We developed both clocked (discrete-time) and unclocked (continuous-time) NLLs and studied their behavior by simulation.

Our results are the following. The simulations of the discrete-time system show that $P_{Synch}(k)$ tends asymptotically to a value $P_0 \leq 1$. The lower the signal-to-noise ratio (SNR), the smaller P_0. In addition the rise of $P_{Synch}(k)$ depends on the SNR as well as on the length of the LFSR: the lower the SNR or the longer the LFSR, the smaller the rise. The same holds for the TUA, i.e. the lower the SNR or the larger the LFSR length, the larger the TUA.

The simulations of the continuous-time system show that the MSE decreases monotonically as a function of time until it reaches a value $\epsilon \geq 0$. The lower the signal-to-noise ratio (SNR), the larger ϵ becomes and the slower the MSE decreases.

The NLL is just in its infancy, especially the unclocked NLL. However, we have shown that the design of both clocked and unclocked NLLs is theoretically possible. The natural next step is the implemention of such systems in hardware.

Appendix A

Proof of Theorem 3.1

Theorem A.1 *Assuming the probalities* $p\left(X_i\right)$ *and* $p\left(X_j\right)$ *to be* $\frac{1}{2}$*, the outputs of the Soft Limiters* $p\left(X_i = 1\right)$ *and* $p\left(X_j = 1\right)$ *can be written as*

$$p\left(X_i = 1\right) = \frac{1}{2}\left(1 + \frac{m_{Filter1}(t)}{\kappa_1(m_{Filter1})}\right) \tag{A.1}$$

$$p\left(X_j = 1\right) = \frac{1}{2}\left(1 + \frac{m_{Filter2}(t)}{\kappa_2(m_{Filter2})}\right) \tag{A.2}$$

where

$$\kappa_1(m) = \frac{\mathrm{E}_m\left[g_{==}(\theta)\right] + \mathrm{E}_m\left[g_{=\neq}(\theta)\right] + \mathrm{E}_m\left[g_{\neq=}(\theta)\right] + \mathrm{E}_m\left[g_{\neq\neq}(\theta)\right]}{4} \tag{A.3}$$

$$\kappa_2(m) = \frac{\mathrm{E}_m\left[q_{==}(\theta)\right] + \mathrm{E}_m\left[q_{=\neq}(\theta)\right] + \mathrm{E}_m\left[q_{\neq=}(\theta)\right] + \mathrm{E}_m\left[q_{\neq\neq}(\theta)\right]}{4} \tag{A.4}$$

with

$$\theta = t - \tau \tag{A.5}$$

and

$$\mathrm{E}_m\left[g(\theta)\right] = \int_{D_1(m)} g(\theta) f(\theta)\,\mathrm{d}\theta \tag{A.6}$$

$$D_1(m) = \left\{\theta \in [-T/2, T/2] \;\middle|\; g(\theta) \geq |m|\right\} \tag{A.7}$$

resp.

$$\mathrm{E}_m\left[q(\theta)\right] = \int_{D_2(m)} q(\theta) f(\theta)\,\mathrm{d}\theta \tag{A.8}$$

$$D_2(m) = \left\{\theta \in [-T/2, T/2] \;\middle|\; q(\theta) \geq |m|\right\} \tag{A.9}$$

where $f(\theta)$ *denotes the probability density function of* θ*.*

Let $X_i \in \{1, -1\}$ be a stochastic variable with probability $p(x)$. The signal after *Filter 1* or *Filter 2* can be written as:

$$x(t) = \sum_i X_i g_{\tilde{X}_{i-1}\tilde{X}_{i+1}}(t - iT - \tau) \tag{A.10}$$

where the waveform $g_{\tilde{X}_{i-1}\tilde{X}_{i+1}}(t) = 0 \ \forall |t| > T/2$. Note that the waveform depends on X_{i-1}, X_i and X_{i+1}.

Without loss of generality we will assume that $i = 0$. So (A.10) can be written as

$$x(t) = X_0 g_{\tilde{X}_{-1}\tilde{X}_{+1}}(t - \tau) \tag{A.11}$$

τ is uniformly distributed in $\Delta(m) = \Big\{\tau \in [t - T/2, t + T/2] \mid g_{\tilde{X}_{-1}\tilde{X}_{+1}}(t-\tau) \geq |m|\Big\}$ with $m = \mathrm{E}\left[x(t)\right]$. $g(t-\tau) \geq |m|$ because

$$\begin{aligned} |m| &= |\mathrm{E}\left[x(t)\right]| \\ &= \Big|\mathrm{E}\Big[X_0 g_{\tilde{X}_{-1}\tilde{X}_{+1}}(t-\tau)\Big]\Big| \\ &= \Big|\mathrm{E}\left[X_0\right]\mathrm{E}\Big[g_{\tilde{X}_{-1}\tilde{X}_{+1}}(t-\tau)\Big]\Big| \\ &\leq \Big|X_0 \mathrm{E}\Big[g_{\tilde{X}_{-1}\tilde{X}_{+1}}(t-\tau)\Big]\Big| \\ &= |X_0| \Big|g_{\tilde{X}_{-1}\tilde{X}_{+1}}(t-\tau)\Big| \\ &= g_{\tilde{X}_{-1}\tilde{X}_{+1}}(t-\tau) \end{aligned} \tag{A.12}$$

where we use the fact that

$$\begin{aligned} \mathrm{E}\left[X_o\right] &= p\left(X_0 = 1\right) - p\left(X_0 = -1\right) \\ &= 2p\left(X_0 = 1\right) - 1 \\ &\leq 1 \end{aligned} \tag{A.13}$$

In order to design the Soft Limiter in subsection 3.2.2 we need to know $p\left(x_0\right)$ as a function of m. So we have

$$m = \mathrm{E}\left[x(t)\right] \tag{A.14}$$

$$= \mathrm{E}\Big[X_0 g_{\tilde{X}_{i-1}\tilde{X}_{i+1}}(t-\tau)\Big] \tag{A.15}$$

$$= \int_{\Delta(m)} \sum_{x_{-1},x_0,x_{+1}} p\left(x_{-1}, x_0, x_{+1}\right) p(\tau) x_0 g_{\tilde{X}_{-1}\tilde{X}_{+1}}(t-\tau)\,\mathrm{d}\tau \tag{A.16}$$

$$= \sum_{x_{-1},x_0,x_{+1}} p\left(x_{-1}, x_0, x_{+1}\right) x_0 \int_{\Delta(m)} p(\tau) g_{\tilde{X}_{-1}\tilde{X}_{+1}}(t-\tau)\,\mathrm{d}\tau \tag{A.17}$$

In (A.16) we used the fact that τ is independent of X_{-1}, X_0 and X_{+1}. We substitute $t-\tau$ by θ so we get

$$\begin{aligned} m &= \sum_{x_{-1},x_0,x_{+1}} p\left(x_{-1},x_0,x_{+1}\right) x_0 \int_{D(m)} p(\theta) g_{\tilde{X}_{-1}\tilde{X}_{+1}}(\theta)\,\mathrm{d}\theta && \text{(A.18)} \\ &= \sum_{x_{i-1},x_0,x_{+1}} p\left(x_{-1}\right) p\left(x_0\right) p\left(x_{+1}\right) x_0 \mathrm{E}_m\left[g_{\tilde{X}_{-1}\tilde{X}_{+1}}(\theta)\right] && \text{(A.19)} \\ &= p\left(X_0=1\right)\Big[p\left(X_{-1}=1\right)p\left(X_{+1}=1\right)\mathrm{E}_\theta\left[g_{==}(\theta)\right] \\ &\quad +p\left(X_{-1}=1\right)p\left(X_{+1}=-1\right)\mathrm{E}_m\left[g_{=\neq}(\theta)\right] \\ &\quad +p\left(X_{-1}=-1\right)p\left(X_{+1}=1\right)\mathrm{E}_m\left[g_{\neq=}(\theta)\right] \\ &\quad +p\left(X_{-1}=-1\right)p\left(X_{+1}=-1\right)\mathrm{E}_m\left[g_{\neq\neq}(\theta)\right]\Big] \\ &\quad -p\left(X_0=-1\right)\Big[p\left(X_{-1}=1\right)p\left(X_{+1}=1\right)\mathrm{E}_m\left[g_{\neq\neq}(\theta)\right] \\ &\quad +p\left(X_{-1}=-1\right)p\left(X_{+1}=1\right)\mathrm{E}_m\left[g_{=\neq}(\theta)\right] \\ &\quad +p\left(X_{-1}=1\right)p\left(X_{+1}=-1\right)\mathrm{E}_m\left[g_{\neq=}(\theta)\right] \\ &\quad +p\left(X_{-1}=-1\right)p\left(X_{+1}=-1\right)\mathrm{E}_m\left[g_{==}(\theta)\right]\Big] && \text{(A.20)} \end{aligned}$$

With $D(m) = \left\{\theta \in [-T/2, T/2] \;\middle|\; g_{\tilde{X}_{i-1}\tilde{X}_{i+1}}(\theta) \geq |m|\right\}$. (A.19) follows because X_{-1}, X_0 and X_{+1} are independent.

By using (A.20) and the fact that $p\left(X_0=-1\right) = 1-p\left(X_0=1\right)$ we get

$$p\left(X_0=1\right) = \frac{m+\kappa(m)}{\Pi_{=}\Big(\mathrm{E}_m\left[g_{==}(\theta)\right]+\mathrm{E}_m\left[g_{\neq\neq}(\theta)\right]\Big)+\Pi_{\neq}\Big(\mathrm{E}_m\left[g_{=\neq}(\theta)\right]+\mathrm{E}_m\left[g_{\neq=}(\theta)\right]\Big)} \quad \text{(A.21)}$$

with

$$\begin{aligned} \Pi_{=} &= p\left(X_{-1}=1\right)p\left(X_{+1}=1\right) \\ &\quad +p\left(X_{-1}=-1\right)p\left(X_{+1}=-1\right) && \text{(A.22)} \\ \Pi_{\neq} &= p\left(X_{-1}=1\right)p\left(X_{+1}=-1\right) \\ &\quad +p\left(X_{-1}=-1\right)p\left(X_{+1}=+1\right) && \text{(A.23)} \\ \kappa(m) &= p\left(X_{-1}=1\right)p\left(X_{+1}=1\right)\mathrm{E}_m\left[g_{\neq\neq}(\theta)\right] \\ &\quad +p\left(X_{-1}=1\right)p\left(X_{+1}=1\right)\mathrm{E}_m\left[g_{=\neq}(\theta)\right] \\ &\quad +p\left(X_{-1}=1\right)p\left(X_{+1}=-1\right)\mathrm{E}_m\left[g_{\neq=}(\theta)\right] \\ &\quad +p\left(X_{-1}=-1\right)p\left(X_{+1}=-1\right)\mathrm{E}_m\left[g_{==}(\theta)\right] && \text{(A.24)} \end{aligned}$$

Assuming that

$$\begin{array}{rclcl} p\left(X_{-1}=1\right) &=& p\left(X_{-1}=-1\right) &=& \frac{1}{2} \\ p\left(X_{+1}=1\right) &=& p\left(X_{+1}=-1\right) &=& \frac{1}{2} \end{array} \quad \text{(A.25)}$$

we get

$$\Pi_{=} = \frac{1}{2} \tag{A.26}$$

$$\Pi_{\neq} = \frac{1}{2} \tag{A.27}$$

$$\kappa(m) = \frac{\mathrm{E}_m\left[g_{==}(\theta)\right]+\mathrm{E}_m\left[g_{=\neq}(\theta)\right]+\mathrm{E}_m\left[g_{\neq=}(\theta)\right]+\mathrm{E}_m\left[g_{\neq\neq}(\theta)\right]}{4} \tag{A.28}$$

By using (A.26), (A.27) and (A.28), $p\left(X_0 = 1\right)$ becomes

$$p\left(X_0 = 1\right) = \frac{1}{2}\left(1 + \frac{m}{\kappa(m)}\right) \tag{A.29}$$

$\square$

Bibliography

[1] John G. Proakis. *Digital Communications.* McGraw-Hill, 2001.

[2] Hans-Andrea Loeliger. Stochastische Modelle und Signalverarbeitung. Lecture notes, Signal and Information Processing Laboratory, ETH Zurich, 2001/02.

[3] A. J. Viterbi. *CDMA, Principles of Spread Spectrum Communication.* Reading, MA: Addison-Wesley Longman Inc., 1995.

[4] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory.* Wiley Series in Telecommunication, 1991.

[5] Felix Lustenberger. *On the design of analog VLSI interative decoders.* PhD thesis, Swiss Federal Institute of Technology, Zurich, 2000.

[6] Norbert R. Malik. *Electronic Circuits: Analysis, Simulation and Design.* Prentice Hall, 1995.