Bachelor Project

Czech Technical University in Prague

F3

Faculty of Electrical Engineering
Department of radio engineering

Implementation of the Communication Physical Layer in the over-the-air Test Bed

Michael Kimmer

Supervisor: prof. Ing. Jan Sýkora, CSc.
May 2022

BACHELOR‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Kimmer Michael** Personal ID number: **491909**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Radioelectronics**

Study program: **Open Electronic Systems**

II. Bachelor’s thesis details

Bachelor’s thesis title in English:

Implementation of the Communication Physical Layer in the over-the-air Test Bed

Bachelor’s thesis title in Czech:

Implementace fyzické vrstvy komunikačního systému v rádiové testovací platformě

Guidelines:

Student will first get acquainted with fundamentals of coding and synchronisation technique for radio communication systems. The goal of the work is to design, analyse and functionally verify selected modulation, coding and synchronisation methods in the over-the-air test bed. This should include the pseudo real time implementation of properly modularised algorithms in the USRP Ettus based test bed. Next, if possible, should be designed a set of custom HW/RF/DSP dedicated blocks for true real-time implementation of the communication physical layer processing chain on a suitable selected HW platform and with possible utilisation of market-available RF modules. A special attention should be given to possible utilisation of the system as laboratory tools in digital communication and signal processing related courses taught at the faculty.

Bibliography / sources:

[1] J. G. Proakis: Digital communications, 4th ed. 2001
[2] Ettus Research [cit. 2020-01-27], https://www.ettus.com/

Name and workplace of bachelor’s thesis supervisor:

prof. Ing. Jan Sýkora, CSc. Department of Radioelectronics FEE

Name and workplace of second bachelor’s thesis supervisor or consultant:

Date of bachelor’s thesis assignment: **27.01.2022** Deadline for bachelor thesis submission: **20.05.2022**

Assignment valid until: **30.09.2023**

prof. Ing. Jan Sýkora, CSc. Supervisor’s signature	doc. Ing. Stanislav Vítek, Ph.D. Head of department’s signature	prof. Mgr. Petr Páta, Ph.D. Dean’s signature

III. Assignment receipt

The student acknowledges that the bachelor's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the bachelor’s thesis, the author must state the names of consultants and include a list of references.

. Date of assignment receipt

Student’s signature

Acknowledgements

I would like to thank my supervisor, prof. Ing. Jan Sýkora, CSc. for his guidance and valuable advice. Next, I would like to thank my family for supporting me during my studies, and my friend Elizabeth Horynová for the thesis language correction.

Declaration

I hereby declare that the work has been done by myself independently and that I have mentioned all the sources used in accordance with methodical instruction about adhering ethical principles when preparing university final works.

In Prague, May 20, 2022

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne 20. 5. 2022

. . . . . .

Abstract

The goal of this thesis is to get acquainted with the basics of the over-the-air digital communication on the *physical layer*. For this purpose a simple transmitter with *convolutional coder*, *QPSK modulator*, channel model and receiver with *Viterbi decoder* are algorithmically designed in program *MATLAB*. Emphasis is placed on synchronization of frequency offset and delay. Then the channel simulator is replaced with a real transmitter and receiver pair. For this objective, we use the *USRP N210* software defined radios from the company *Ettus Research™*. [5]

Keywords: digital communication, convolutional code, digital modulation, QPSK, synchronization, Zadoff–Chu sequence, Pseudorandom-Noise sequence, Viterbi algorithm, SDR

Supervisor: prof. Ing. Jan Sýkora, CSc.

Abstrakt

Cílem této práce je seznámit se se základy bezdrátové digitální komunikace na *fyzické vrstvě*. Tyto znalosti jsou ověřeny algoritmickým návrhem vysílače s *konvolučním kodérem* a *QPSK modulací*, modelu kanálu a přijímače s *Viterbiho dekodérem* v programu *MATLAB*. Důraz je kladen na synchronizaci zpoždění a frekvenčního offsetu. Následně je model kanálu nahrazen reálným vysílačem a přijímačem, k tomuto účelu použijeme *USRP N210* softwarově definovaná rádia od firmy *Ettus Research™*. [5]

Klíčová slova: digitální komunikace, konvoluční kód, digitální modulace, QPSK, synchronizace, Zadoff–Chu sekvence, Pseudorandom-Noise sekvence, Viterbiho algoritmus, SDR

Překlad názvu: Implementace fyzické vrstvy komunikačního systému v rádiové testovací platformě

Contents

1 Introduction 1

2 Fundamentals of digital communication 3

2.1 System model 3
2.2 Coding 3
2.2.1 Convolutional codes 4
2.3 Digital modulation 4
2.3.1 Linear memoryless modulations (1D) 5
2.3.2 Nyquist condition 5
2.3.3 Discrete samples realization . . 7
2.4 Communication channel 7
2.4.1 Stochastic AWGN channel description 8
2.5 Channel parameters estimation . 10
2.5.1 Cramér–Rao lower bound . . . 11
2.5.2 Signal detection 11
2.6 Demodulation and decoding 12
2.6.1 Viterbi algorithm 13

3 Theory application 15

3.1 Parameters estimation 15
3.1.1 Delay estimation 15
3.1.2 Frequency offset estimation . . 18
3.1.3 Amplitude attenuation estimation 18
3.1.4 Phase estimation 19
3.2 Signal detection 19
3.3 CRLB of the frequency offset . . . 21

4 Implemented models 23

4.1 System blocks implementation . . 23
4.1.1 Convolutional code 23
4.1.2 QPSK modulation 23
4.1.3 Pilot signals 24
4.1.4 AWGN channel simulation . . 25
4.1.5 Parameters estimation and signal detection 25
4.1.6 Signal decomposition and equalization 28
4.1.7 Viterbi decoder 29
4.2 MATLAB simulation 31
4.3 SDR implementation 34
4.3.1 Ettus USRP N210 SDR 34
4.3.2 Data transmission using SDRs 35

5 Conclusion 39

Bibliography 41

A Attached files 43

Figures

2.1 Block diagram of the system model 3
2.2 Root raised cosine pulse [6] 7
2.3 BPSK and QPSK decision regions 13

3.1 Zadoff–Chu sequence and its cross-correlation 17
3.2 PN sequence generator[1, p. 797] 17

4.1 Cross-correlation of used modulated Zadoff–Chu sequence .. 26
4.2 Cross-correlation of used modulated Pseudorandom-Noise sequence 27
4.3 Transitions in modulator states . 29
4.4 Comparison of sent and received data (zoomed) 32
4.5 Used RRC pulse and modulated signal 32
4.6 Estimation of delay 32
4.7 Estimation of frequency offset .. 33
4.8 Phase estimation 33
4.9 Decomposed received signal in constellation space 33
4.10 PSD with measured frequency offset 35
4.11 Estimation of delay (true) 36
4.12 Phase estimation (true) 36
4.13 Received bitmap image 37

Tables

4.1 Computed CRLB of f_o standard deviations 28
4.2 Codeword and QPSK channel symbol of each transition 30

Chapter 1

Introduction

The task of communication is generally to get data from one place to another. The need of communication is very high among people and an effort for a fast and reliable communication gave birth to today's interconnected world of Internet, mobile networks and fast electronic devices.

Most communication nowadays is digital thanks to its high precision and possibility to use advanced coding, error detection and repairing algorithms. Digital data denotes the data which is discrete both in time and value, mostly bit sequences.

The goal of this thesis is to get acquainted with the basics of the over-the-air digital communication on the *physical layer*. For this purpose we will algorithmically design a simple transmitter, a channel simulator and a receiver with a signal parameters estimator for a chosen combination of methods in the program *MATLAB*. Then the channel simulator will be replaced with a real transmitter and receiver. For this objective we will use the *Universal Software Radio Peripheral* (*USRP 210*) software defined radio (SDR) from the company *Ettus Research*™. [5]

Chapter 2

Fundamentals of digital communication

2.1 System model

The digital communication model generally consists of a coder, modulator, channel, channel parameters estimator, demodulator or signal basis decomposer and decoder. In this model we use a linear channel with *additive white gaussian noise* (*AWGN*) which already contains signal conversion from the complex envelope to the real signal, real channel model and conversion back to the complex envelope (this is usually done in the analog part of hardware).

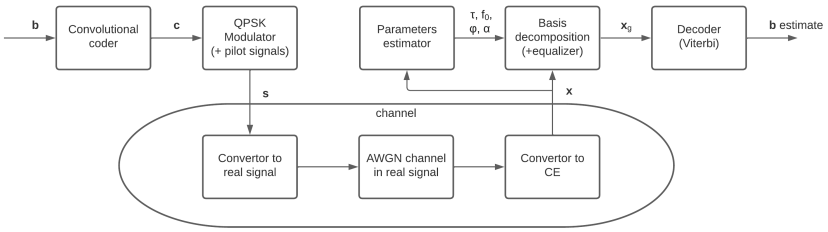


Figure 2.1: Block diagram of the system model

2.2 Coding

In order to attain better error performance of digital communication (over noisy channels), the input data $\boldsymbol{b}$ are modified by appropriately adding redundant information, this process is called coding. The degree of redundancy in the original data can be indicated by the *code rate* R_c, it is defined as the ratio between dimensions of input k-bit data elements and encoded n-bit elements, called codewords. [1, p. 2]

$$R_c = \frac{k}{n} \tag{2.1}$$

Most used classical codes are linear codes, convolutional codes and trellis codes. In these days they are largely replaced in more powerful applications

by more sophisticated techniques such as *turbocodes*, *polar codes* or *LDPC codes*. [7] Next we will discuss only convolutional codes since they are easy to implement and their performance is close to channel capacity in many cases. [1, p. 491]

2.2.1 Convolutional codes

Convolutional codes over binary data can be easily implemented by passing the input data $\boldsymbol{b}_n$ into shift registers of length K (called the *constraint length* of the convolutional code). The elements of the actual output codeword element $\boldsymbol{c}_n$ are then computed as linear combinations of the elements of the actual input data and the older data from the registers. This process can be generalized over different fields, however, we will continue over the Galois field GF(2). Rewritten to the matrix form it gives the formula

$$\boldsymbol{c}_n = \sum_{i=0}^{K} \boldsymbol{G}_i \boldsymbol{b}_{n-i} \tag{2.2}$$

similar to the convolution where $\boldsymbol{G}_i$ are generator matrices.

2.3 Digital modulation

Digital data, mostly bit sequences, cannot be transmitted through the space in their original form because space and materials show continuous physical properties. They must first be appropriately converted to the continuous signal which can travel through the particular medium and can later be transformed back to the original digital data. This conversion is called digital modulation. Our goal is to construct it to be reliable and fast.

Generally, most of modulations can be written as a sum of modulation pulses $\boldsymbol{g}$ shifted in time (in multiplies of symbol periods T_s), dependent on the actual (coded) data symbol $\boldsymbol{c}_n$ and on all previous data symbols contained in the modulator state $\boldsymbol{\sigma}_n$. Dependency on the data can be summarized by creating a new complex-valued variable called *channel symbol* $\boldsymbol{q}_n$.

$$s(t) = \sum_n \boldsymbol{g}(\boldsymbol{c}_n, \boldsymbol{\sigma}_n, t - nT_s) = \sum_n \boldsymbol{g}(\boldsymbol{q}_n, t - nT_s) \tag{2.3}$$

$$\boldsymbol{q}_n = \boldsymbol{q}(\boldsymbol{c}_n, \boldsymbol{\sigma}_n) \tag{2.4}$$

There exist some exceptions: for example, the sum can be replaced by the product (CPM modulation). [2]

Digital modulations can be divided according to their mathematical properties. The main division can be done with regard to linearity and memory. Linearity in linear modulations defines pulse $\boldsymbol{g}(\boldsymbol{q}_n, t - nT_s)$ dependence on the channel symbol $\boldsymbol{q}_n$ to be linear. Absence of memory removes dependence on the modulator state $\boldsymbol{\sigma}_n$ from $\boldsymbol{q}(\boldsymbol{c}_n, \boldsymbol{\sigma}_n)$; thus every channel symbol (and so pulse) depends only on the actual data symbol $\boldsymbol{c}_n$. Memory included in modulation can be used for error correction and signal properties improvement;

however, an important part of the memory in the modulator can be replaced by the memory added beforehand in coding. [2] From this point on, we will consider only one-dimensional (1D) modulations (scalar g, σ_n and q_n).

2.3.1 Linear memoryless modulations (1D)

Linearity implies that a modulation consists only of shifted copies of one pulse $g(t - nT_s)$, each multiplied by the respective *channel symbol* q_n which (from the memoryless property) depends only on the actual data symbol $\boldsymbol{c}_n$. As a result we get the simplest form of digital modulation. The formula can be written as

$$s(t) = \sum_n q_n g(t - nT_s) \tag{2.5}$$

$$q_n = q(\boldsymbol{c}_n) \tag{2.6}$$

The particular linear modulation is defined by the set of its possible values of channel symbols q_n in Equation 2.6.

If we consider q_n to be only real numbers, we define the set of all possible q_n of $M_q = 2^k$ values as

$$q_n \in \{\pm 1, \pm 2, \ldots, \pm(M_q - 1)\} \tag{2.7}$$

This modulation is called *pulse amplitude modulation* (*PAM*).

Another approach is to make the absolute value of q_n constant and discern only its phase, thus

$$q_n \in \{e^{j\frac{2\pi}{M_q}i}\}_{i=0}^{M_q-1} \tag{2.8}$$

This modulation is called *phase shift keying* (*PSK*) or *MPSK* where $M = M_q$ denotes number of possible constellation points (2PSK is often called *BPSK* and 4PSK shifted by $\frac{\pi}{4}$ *QPSK*). It should be emphasised the energy of each symbol is identical.

The combination of these two approaches gives an often used modulation called *quadrature amplitude modulation* (*QAM*) wherein channel symbols can lie in rectangular grid intersections.

It should be pointed out that the mean value of the channel symbols should be zero. This minimizes the mean symbol energy (and transmitted power) while keeping the difference energy between the sent pulses.

2.3.2 Nyquist condition

In order to subsequently demodulate symbols, we strive to avoid the inter-symbol interference (*ISI*). It can be achieved by making all sent symbols in different times mutually orthogonal. This property is guaranteed by the Nyquist condition. From this definition we can write its basic form.

$$\mathcal{R}^{\mathcal{E}}{}_{g_1,g_2}[m-n] = \int_{-\infty}^{\infty} g_1(t - nT_s) g_2^*(t - mT_s) \mathrm{d}t = 0, \forall n \neq m \tag{2.9}$$

Nyquist condition can be then derived in the frequency domain

$$\sum_n \mathcal{S}^{\mathcal{E}}{}_{g_1,g_2}(f - \frac{n}{T_s}) = T_s \mathcal{R}^{\mathcal{E}}{}_{g_1,g_2}[0], \forall f \tag{2.10}$$

where $\mathcal{R}^{\mathcal{E}}{}_{g_1,g_2}$ represents the pulses' correlation function and $\mathcal{S}^{\mathcal{E}}{}_{g_1,g_2}$ the power spectral density (PSD).[2]

For linear modulations where only one pulse $g(t)$ is used, this formula gets simpler as $g_1(t) = g_2(t) = g(t)$. This condition strictly restricts the number of usable pulses (orthogonal to their shifted copies). At the same time it gives us an easy method for their construction. It states that the orthogonal pulse must have a constant sum of its shifted PSDs.

The easiest way to create an orthogonal pulse is to divide this constant into rectangular functions which represent those particular shifted PSDs. Thus the PSD of this pulse is

$$\mathcal{S}^{\mathcal{E}}{}_g(f) = \begin{cases} T_s \mathcal{R}^{\mathcal{E}}{}_g[0], & |f| \leq \frac{1}{2T_s} \\ 0, & \text{otherwise} \end{cases} \tag{2.11}$$

and the pulse can be for example:

$$g(t) = \frac{\sin(\frac{\pi t}{T_s})}{\frac{\pi t}{T_s}} = \text{sinc}(\frac{t}{T_s}) \tag{2.12}$$

Such pulse is noncausal so first it must be cut and delayed, however it decays as $1/t$ thus the delay must be large to cover the pulse's energy. Moreover, a small mistiming error in sampling at the demodulator will result in an infinite series of ISI components whose sum does not converge. [1, p. 607]

Another pulse widely used in digital communications fulfilling this condition is the *root raised cosine pulse* (RRC). It has been constructed with the goal to suppress the shortcomings from the first pulse. It decays as $1/t^3$ and its sum of ISI components converges to a finite value. It is defined in spectrum as

$$\mathcal{F}_g(f) = \begin{cases} \sqrt{T_s}, & |f| < \frac{1-\alpha}{2T_s} \\ \sqrt{T_s}\cos(\frac{\pi T_s}{2\alpha}(|f| - \frac{1-\alpha}{2T_s})), & \frac{1-\alpha}{2T_s} \leq |f| < \frac{1+\alpha}{2T_s} \\ 0, & \frac{1+\alpha}{2T_s} \leq |f| \end{cases} \tag{2.13}$$

where $\alpha \in <0, 1>$ is the *roll-off factor* that describes the frequency overlaps. In the time domain this pulse can be described by

$$g(t) = \frac{1}{\sqrt{T_s}(1 - 16\alpha^2 \frac{t^2}{T_s^2})}(\frac{\sin((1-\alpha)\frac{\pi t}{T_s})}{\frac{\pi t}{T_s}} + \frac{4\alpha\cos((1+\alpha)\frac{\pi t}{T_s})}{\pi}) \tag{2.14}$$

(singular points at $t \in \{0, \pm\frac{T_s}{4\alpha}\}$ can be defined by limit). [2]

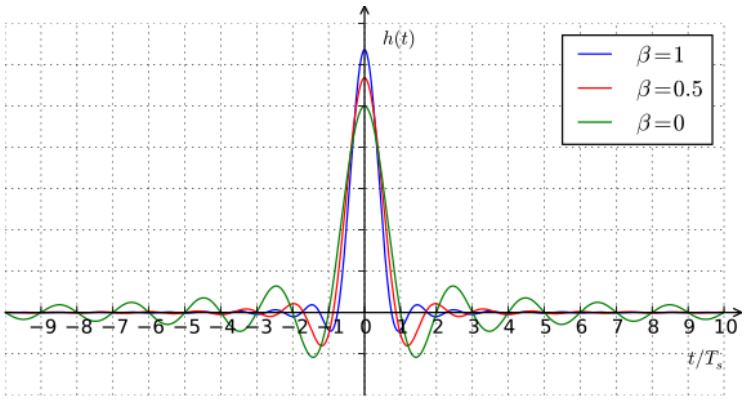


Figure 2.2: Root raised cosine pulse [6]

2.3.3 Discrete samples realization

Digital signal processing done on microprocessors is performed discretely in time on signal samples since the transformation to the continuous voltage is done as one of last procedures. Because of this, it is necessary to implement the modulation on signal samples.

The goal is to get the modulated signal samples $s[k] = s(kT_p)$ on the sampling period Tp from samples of the modulation pulse $g[k] = g(kT_p)$ and channel symbols q_n. We start by substituting from Equation 2.5

$$s[k] = s(kT_p) = \sum_n q_n g(kT_p - nT_s) \tag{2.15}$$

For simplicity, the symbol period is usually chosen as an integer multiple of the sampling period. This ratio is denoted N_s.

$$T_s = N_s T_p \tag{2.16}$$

After substitution we get

$$s[k] = \sum_n q_n g(kT_p - nN_s T_p) = \sum_n q_n g[k - nN_s] \tag{2.17}$$

Now, as $g[k-m] = \delta[k-m] * g[k]$ we can write

$$s[k] = \sum_n q_n \delta[k - nN_s] * g[k] = (\sum_n q_n \delta[k - nN_s]) * g[k] \tag{2.18}$$

From this we can conclude that the modulation samples can be easily computed as the convolution of upsampled modulation symbols and the modulation pulse.

2.4 Communication channel

The communication channel is generally any form of media which can connect the transmitter and the receiver; it may be a pair of wires, or an optical fiber,

or an underwater channel with an acoustical wave transmission, or any storage media. [1, p. 3] In our case the channel will be an environment capable of electromagnetic wave propagation, specifically the air, in combination with the hardware of the communication system.

Besides apparent parameters, such as time delay and signal amplitude attenuation, communication channels suffer from additive noise. The additive noise is usually produced by components of the communication system, but it also comes from the environment. The effects of noise may be minimized by increasing the power of the transmitted signal and by the appropriate design of the signal and its demodulator. [1, p. 3, 4]

The simplest usable complete mathematical description of a communication channel is the additive white gaussian noise (AWGN) channel with constant signal delay τ, the amplitude attenuation α and the frequency offset f_o of oscillators. The complex AWGN we denote by $w(t)$. Since we work on the complex envelope of the real transmitted signal, we have to consider another parameter, the complex angular rotation (or the phase) φ. Although this parameter depends only on the signal delay τ and the carrier frequency f_c (including f_o), it cannot be computed from these thanks to high carrier frequencies which make enormous phase estimation errors from negligible delay estimation errors. [1, p. 290] The resulting mathematical model can be thus expressed as

$$x(t) = \alpha e^{\mathrm{j}\varphi} e^{\mathrm{j}2\pi f_o(t-\tau)} s(t-\tau) + w(t) \tag{2.19}$$

where $s(t)$ is the sent signal and $x(t)$ the received signal.

2.4.1 Stochastic AWGN channel description

White gaussian noise

Complex white gaussian noise in the complex envelope $w(t)$ is the white gaussian process in its both real and imaginary components. The white property defines constancy of its power spectral density $\mathcal{S}_w = 2N_0$ so it can be characterized by the *noise spectral density* N_0 (or by variance σ_w) and probability density function (*PDF*).

$$p_w(w) = \frac{1}{\pi\sigma_w^2} \exp(-\frac{|w|^2}{\sigma_w^2}) \tag{2.20}$$

Dependency only on the absolute value of the noise's value implies the noise to be rotationally invariant, thus the noise does not change properties after multiplication by any complex exponential. Its constant power spectral density is often measured relatively to signal power as the *signal-to-noise ratio* (*SNR*) or normalised to E_b/N_0, where E_b is the mean power needed for transmission of one bit.

Projection of AWGN $w(t)$ into an orthonormal signal basis of the constellation space yields resulting noise in the constellation space $w_{g,n}$. In our case the basis is shifted modulation pulses $g(t - nT_S)$ from subsection 2.3.2. This

noise preserves its gaussian character and zero mean value. Its variance equals the power spectral density of the continuous noise (from Karhunen-Loève expansion). [1, p. 78]

$$\sigma^2_{w_{g,n}} = \mathcal{S}_w = 2N_0 \tag{2.21}$$

Moreover, every element is independent of the others, so for the whole vector the joint PDF can be computed

$$\begin{aligned} p_{\boldsymbol{w}_g}(\boldsymbol{w}_g) &= \prod_n p_{w_{g,n}}(w_{g,n}) = \prod_n \frac{1}{2\pi N_0} \exp(-\frac{|w_{g,n}|^2}{2N_0}) \\ &= \frac{1}{\pi^n (2N_0)^n} \exp(-\frac{\|\boldsymbol{w}_g\|^2}{2N_0}) \end{aligned} \tag{2.22}$$

It should also be mentioned, that the noise samples also have gaussian character, zero mean, but they have a different variance than the noise in the constellation space. Samples can be viewed as coefficients of the decomposition of the continuous noise $w(t)$ into the sampling functions $\mathrm{sinc}(\frac{t-T_p}{T_p})$ shifted by the sampling period T_p

$$\mathrm{sinc}(t) = \frac{\sin(\pi t)}{\pi t} \tag{2.23}$$

$$\sigma^2_{w_k} = \frac{2N_0}{T_p} \tag{2.24}$$

Their joint *PFD* is thus

$$p_{\boldsymbol{w}}(\boldsymbol{w}) = \prod_k p_{w_k}(w_k) = \frac{1}{\pi^n (\frac{2N_0}{T_p})^n} \exp(-\frac{\|\boldsymbol{w}\|^2}{\frac{2N_0}{T_p}}) \tag{2.25}$$

Likelihood function

In search of the unknown input data and channel parameters, we start by describing the channel input-output relation. In the stochastic channel we can use the probability of the channel output samples $\boldsymbol{x}$ conditioned by the knowledge of random and unknown parameters

$$p(\boldsymbol{x}|\boldsymbol{\theta}, \boldsymbol{\omega}, \boldsymbol{w}) \tag{2.26}$$

where $\boldsymbol{\theta}$ denotes the input data (thus also the input signal $\boldsymbol{s}$) and $\boldsymbol{\omega}$ channel parameters α, τ, φ and f_o. Conditioning the output probability by knowledge of all parameters and data can be interpreted as knowledge of the output $\boldsymbol{x}$

$$\boldsymbol{x} = \alpha e^{\mathrm{j}\varphi} \boldsymbol{s}_{\tau, f_o} + \boldsymbol{w} \tag{2.27}$$

This becomes deterministic with known probability.

$$p(\boldsymbol{x}|\boldsymbol{\theta}, \boldsymbol{\omega}, \boldsymbol{w}) = \delta(\boldsymbol{x} - (\alpha e^{\mathrm{j}\varphi} \boldsymbol{s}_{\tau, f_o} + \boldsymbol{w})) \tag{2.28}$$

where $\boldsymbol{s}_{\tau,f_o}$ represents the transmitted signal samples shifted in time by τ and in frequency by f_o.

$$s_{\tau,f_o}[k] = s[k-\tau]e^{\mathrm{j}2pif_oT_p(k-\tau)} \tag{2.29}$$

In the next step we marginalize the conditional probability over the AWGN distribution because in the real applications we are usually not able to determine its random instantaneous value, whereas its PDF is known.

$$\begin{aligned} p(\boldsymbol{x}|\boldsymbol{\theta},\boldsymbol{\omega}) &= \int_{-\infty}^{\infty} p(\boldsymbol{x}|\boldsymbol{\theta},\boldsymbol{\omega},\boldsymbol{w})p_{\boldsymbol{w}}(\boldsymbol{w})\mathrm{d}\boldsymbol{w} \\ &= \int_{-\infty}^{\infty} \delta(\boldsymbol{x}-(\alpha e^{\mathrm{j}\varphi}\boldsymbol{s}_{\tau,f_o}+\boldsymbol{w}))p_{\boldsymbol{w}}(\boldsymbol{w})\mathrm{d}\boldsymbol{w} \\ &= p_{\boldsymbol{w}}(\boldsymbol{x}-\alpha e^{\mathrm{j}\varphi}\boldsymbol{s}_{\tau,f_o}) \end{aligned} \tag{2.30}$$

By substituting Equation 2.25 we obtain the formula for the channel *likelihood function* (*LF*) on the signal samples level

$$\Lambda(\boldsymbol{x},\boldsymbol{\theta},\boldsymbol{\omega}) = p(\boldsymbol{x}|\boldsymbol{\theta},\boldsymbol{\omega}) = c\exp(-\frac{\|\boldsymbol{x}-\alpha e^{\mathrm{j}\varphi}\boldsymbol{s}_{\tau,f_o}\|^2}{\frac{2N_0}{T_p}}) \tag{2.31}$$

where

$$c = \frac{1}{\pi^n(\frac{2N_0}{T_p})^n} \tag{2.32}$$

As we will later search for its maximum position, we can define the channel *log-likelihood function* (*LLF*) which shares the same extrema positions with the *LF*.

$$\Lambda'(\boldsymbol{x},\boldsymbol{\theta},\boldsymbol{\omega}) = -\|\boldsymbol{x}-\alpha e^{\mathrm{j}\varphi}\boldsymbol{s}_{\tau,f_o}\|^2 \tag{2.33}$$

Derivation of LF and LLF on the constellation space level is identical yielding

$$\Lambda_g(\boldsymbol{x}_g,\boldsymbol{\theta},\boldsymbol{\omega}) = c\exp(-\frac{\|\boldsymbol{x}_g-\alpha e^{\mathrm{j}\varphi}\boldsymbol{s}_{g,f_o}\|^2}{2N_0}) \tag{2.34}$$

where

$$c = \frac{1}{\pi^n(2N_0)^n} \tag{2.35}$$

and

$$\Lambda'_g(\boldsymbol{x}_g,\boldsymbol{\theta},\boldsymbol{\omega}) = -\|\boldsymbol{x}_g-\alpha e^{\mathrm{j}\varphi}\boldsymbol{s}_{g,f_o}\|^2 \tag{2.36}$$

2.5 Channel parameters estimation

In order to estimate the channel parameters, maximizing of the channel *likelihood function* from the Equation 2.31 can be used. This approach can be interpreted as finding such channel coefficients, that the received output was the most likely one. This method is called the *maximum likelihood estimator* (*MLE*). It can be derived that MLE is asymptotically unbiased and efficient with the observation length. For a linear channel with AWGN it applies to any observation length. [3]

Estimated parameters and the data could be estimated all at once by marginalizing the LF over transmitted data, however for better results (especially for higher SNR), synchronization sequences called pilot signals are used. [1, p. 317] They determine the signal $\boldsymbol{s}$ in the LF; moreover, they can be selected appropriately so that the channel parameters can be estimated more precisely.

2.5.1 Cramér–Rao lower bound

The theoretical boundary of accuracy of an estimator (of a deterministic parameter) can be computed by *Cramér–Rao lower bound* (*CRLB*). It can be used to assume whether the estimator can be theoretically implemented, and eventually to which maximal precision the real estimator can be tuned.

The theorem states that if the regularity condition

$$\mathrm{E}\left[\frac{\partial \ln p(\boldsymbol{x}|\theta)}{\partial \theta}\right] = 0 \tag{2.37}$$

holds, then for any unbiased estimator of θ, the lower variance boundary of the estimate $\hat{\theta}$ can be determined as

$$\mathrm{var}\left[\hat{\theta}\right] \geq \left(-\mathrm{E}\left[\frac{\partial^2 \ln p(\boldsymbol{x}|\theta)}{\partial \theta^2}\right]\right)^{-1} = \left(\mathrm{E}\left[\left(\frac{\partial \ln p(\boldsymbol{x}|\theta)}{\partial \theta}\right)^2\right]\right)^{-1} \tag{2.38}$$

The proof can be found in [3].

2.5.2 Signal detection

we need to test the hypothesis of signal presence in order to determine whether the estimator supplied the actual parameters of the transmitted signal, or there was no signal received (so the estimate is only random data). For this objective we use the *Neyman-Pearson* theorem, which provides a method of detection with maximized detection probability, given false alarm probability.

Neyman-Pearson theorem

For the given false alarm probability P_{FA}, the binary hypothesis detector maximizes detection probability P_{D} if the decision is the likelihood ratio test

$$\hat{\boldsymbol{s}} = \begin{cases} \boldsymbol{s}^{(0)}, & \Lambda(\boldsymbol{x}) \leq \gamma \\ \boldsymbol{s}^{(1)}, & \Lambda(\boldsymbol{x}) > \gamma \end{cases} \tag{2.39}$$

where $\boldsymbol{s}^{(i)}$ are signals whose presence is tested as hypotheses. The likelihood ratio is

$$\Lambda(\boldsymbol{x}) = \frac{p(\boldsymbol{x}|\boldsymbol{s}^{(1)})}{p(\boldsymbol{x}|\boldsymbol{s}^{(0)})} \tag{2.40}$$

and the threshold γ is defined by relation

$$P_{\mathrm{FA}} = \int_{\boldsymbol{x}:\Lambda(\boldsymbol{x})>\gamma} p(\boldsymbol{x}|\boldsymbol{s}^{(0)})\mathrm{d}\boldsymbol{x} \tag{2.41}$$

The proof can be found in [3].

2.6 Demodulation and decoding

The process of acquiring the transmitted message from the received signal samples $\boldsymbol{x}$ is called either data detection (for uncoded data) or decoding (more sophisticated method for coded data). These are done in the constellation space of the pulse g, therefore in the basis $\{g(t - nT_s)\}_n$. Coefficients of decomposition of $x(t)$ to the basis are computed by the continuous dot product, and they can be approximated by the discrete dot product from the $x(t)$ samples $\boldsymbol{x}$ and the shifted modulation pulse samples $\boldsymbol{g_n}$.

$$x_{g,n} = \int_{-\infty}^{\infty} x(t)g^*(t - nT_s)\mathrm{d}t \approx T_p \boldsymbol{x}\boldsymbol{g}_{\boldsymbol{n}}^H \tag{2.42}$$

Both detection and decoding find the data estimate $\hat{\boldsymbol{b}}$ according to the chosen optimality criterion which can be mathematically described by minimizing the metric $\rho(\boldsymbol{x}, \check{\boldsymbol{b}})$ over every possible message $\check{\boldsymbol{b}}$. The process of calculation of the metric is called the demodulation.

$$\hat{\boldsymbol{b}} = \arg\min_{\check{\boldsymbol{b}}} \rho(\boldsymbol{x}, \check{\boldsymbol{b}}) \tag{2.43}$$

Setting a goal to minimize the mean message error probability leads us to the *Maximum a posteriori estimator*. For equiprobable messages (or messages with unknown probability) this again gives the *Maximum likelihood estimator MLE* with already derived likelihood function in Equation 2.31 where we already know the channel parameters estimate $\hat{\boldsymbol{\omega}}$. [2] As already mentioned maximization of the *LF* equals to maximization of the *LLF*, therefore we choose the metric to be

$$\rho(\boldsymbol{x}, \check{\boldsymbol{b}}) = -\Lambda'(\boldsymbol{x}, \check{\boldsymbol{b}}, \boldsymbol{\omega}) \approx -\Lambda'(\boldsymbol{x}, \check{\boldsymbol{b}}, \hat{\boldsymbol{\omega}}) = \|\boldsymbol{x} - \hat{\alpha}e^{\mathrm{j}\hat{\varphi}}\boldsymbol{s}_{\hat{\tau},\hat{f}_o}\|^2 \tag{2.44}$$

It is similar in the constellation space

$$\rho(\boldsymbol{x}_g, \check{\boldsymbol{b}}) \approx -\Lambda'_g(\boldsymbol{x}_g, \check{\boldsymbol{b}}, \hat{\boldsymbol{\omega}}) = \|\boldsymbol{x}_g - \hat{\alpha}e^{\mathrm{j}\hat{\varphi}}\boldsymbol{s}_{g,\hat{f}_o}\|^2 \tag{2.45}$$

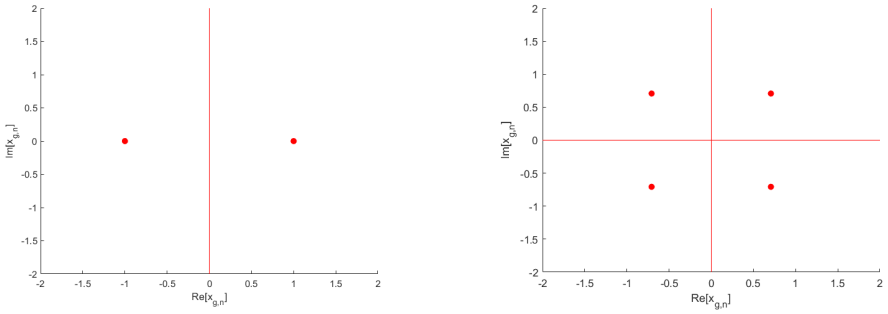
The process of obtaining the message estimate can then be interpreted as choosing the one message whose signal has the minimal error energy to the received signal. As the norm consists of the sum of non-negative elements, the minimization can be perceived element-wise.

$$\rho(\boldsymbol{x}_g, \check{\boldsymbol{b}}) = \sum_n \Delta\rho_{n+1}(x_{g,n}, s_{g,n}) \tag{2.46}$$

$$\Delta\rho_{n+1}(x_{g,n}, s_{g,n}) = |x_{g,n} - \hat{\alpha}e^{\mathrm{j}\hat{\varphi}}s_{g,\hat{f}_o,n}|^2 = |e^{-\mathrm{j}\hat{\varphi}}e^{-\mathrm{j}2\pi\hat{f}_o T_s n}x_{g,n} - \hat{\alpha}q_n|^2 \tag{2.47}$$

Detection is then minimization of Equation 2.46 for uncoded data (this means that the $x_{g,n}$ and q_n are dependent only on the actual b_n). The sum can then be easily minimized by minimization of its elements one by one choosing appropriate b_n (the one with the nearest channel symbol q_n to the equalized $x_{g,n}$) for each received symbol. This operation can be viewed as

dividing the constellation space into *decision regions* by the distance to the nearest possible modulation symbol. Then the symbol estimate $\hat{b}_n$ is decided according to the decision region of the received symbol. For BPSK and QPSK modulations the decision regions are



(a) : BPSK decision regions **(b) :** QPSK decision regions

Figure 2.3: BPSK and QPSK decision regions

Decoding for the coded data has to minimize Equation 2.46 with the difference that each $x_{g,n}$ and q_n depend on the actual b_n and generally on all previous $\{b_i\}_{i=0}^{n-1}$ contained in the modulator state σ_n. Solving such a problem by brute-force computing $\rho(\boldsymbol{x}_g, \check{\boldsymbol{b}})$ for every $\check{\boldsymbol{b}}$ has exponential computational complexity (thus not usable for decoding). Because of this more sophisticated methods for decoding must be used, taking into account the particular structure of the coder.

2.6.1 Viterbi algorithm

The method mostly used for decoding trellis and convolutional codes (with linear computational complexity) is called the *Viterbi algorithm*. It minimizes $\rho(\boldsymbol{x}_g, \check{\boldsymbol{b}})$ by taking into account causality of each q_n and $x_{g,n}(b_n, \sigma_n)$ by minimizing the sum step by step from $n = 0$ (as it was coded) for every possible modulator state. The algorithm consists of two phases.

Recurrent phase

In the first phase, let us consider the step $n = N$ when partial metrics $\rho_N(\boldsymbol{x}_g, \sigma_N)$ for all possible modulator states $\{\sigma_N^{(i)}\}_i$ are already minimized. Then there can be computed new minimized metrics for all modulator states $\{\sigma_{N+1}^{(i)}\}_i$ as

$$\rho_{N+1}(\boldsymbol{x}_g, \sigma_{N+1}) = \min_{\check{\sigma}_N, \check{b}_N} \left[\rho_N(\boldsymbol{x}_g, \check{\sigma}_N) + \Delta\rho_{N+1}(\boldsymbol{x}_g, \check{b}_N, \check{\sigma}_N)\right] \tag{2.48}$$

where $\Delta\rho_{N+1}(\boldsymbol{x}_g, \check{b}_N, \check{\sigma}_N)$ is computed from Equation 2.47 where

$$q_N = q_N(\check{b}_N, \check{\sigma}_N) \tag{2.49}$$

are computed from the particular coding and modulation. In this process there are also stored previous modulator states for each state.

Forward phase

The second phase starts at the last state $\sigma_{N_{\text{end}}}$ (either the one with the minimal metric or the one determined by a given ending *flush sequence*). Then according to the saved previous modulator states the optimal path leading to the last state with the minimal metric is found. Knowing the optimal path, each data estimate $\hat{b}_n$ can be separated from the transitions of modulator states.

Chapter 3

Theory application

3.1 Parameters estimation

In order to estimate the parameters from section 2.5, we can maximize the LF directly by finding its maximum over all parameters. However, it considerably increases the computational complexity, therefore we strive to divide the problem, for example, by searching only for a particular subset of parameters at a time.

3.1.1 Delay estimation

In estimation of the signal delay we start by marginalizing the LF from Equation 2.31 over φ, assuming the uniform distribution in the interval $\varphi \in [0, 2\pi)$.

$$\begin{aligned} p(\boldsymbol{x}|\boldsymbol{s}, \tau, f_o, \alpha) &= \int_{-\infty}^{\infty} p(\boldsymbol{x}|\boldsymbol{s}, \tau, f_o, \alpha, \varphi) p(\varphi|\boldsymbol{s}, \tau, f_o, \alpha) \mathrm{d}\varphi \\ &= \int_0^{2\pi} p(\boldsymbol{x}|\boldsymbol{s}, \tau, f_o, \alpha, \varphi) \frac{1}{2\pi} \mathrm{d}\varphi \\ &= \frac{1}{2\pi} \int_0^{2\pi} c \exp\left(-\frac{\|\boldsymbol{x} - \alpha e^{\mathrm{j}\varphi} \boldsymbol{s}_{\tau, f_o}\|^2}{\frac{2N_0}{T_p}} \right) \mathrm{d}\varphi \end{aligned} \tag{3.1}$$

As the time and frequency shifts do not change the norm of pilot signals $\boldsymbol{s}$, and $\boldsymbol{x}$ is the received signal (thus constant to wanted parameters), we can write

$$\begin{aligned} & p(\boldsymbol{x}|\boldsymbol{s}, \tau, f_o, \alpha) \\ &= \frac{c}{2\pi} \exp\left(-\frac{\|\boldsymbol{x}\|^2 + \alpha^2 \|\boldsymbol{s}\|^2}{\frac{2N_0}{T_p}} \right) \int_0^{2\pi} \exp\left(\frac{2\alpha \Re\left[\langle \boldsymbol{x}, \boldsymbol{s}_{\tau, f_o} \rangle e^{-\mathrm{j}\varphi} \right]}{\frac{2N_0}{T_p}} \right) \mathrm{d}\varphi \\ &= c_2(\alpha) \int_0^{2\pi} \exp\left(\frac{2\alpha |\langle \boldsymbol{x}, \boldsymbol{s}_{\tau, f_o} \rangle| \cos\left[\arg \langle \boldsymbol{x}, \boldsymbol{s}_{\tau, f_o} \rangle - \varphi \right]}{\frac{2N_0}{T_p}} \right) \mathrm{d}\varphi \\ &= c_2(\alpha) \mathrm{I}_0\left(\frac{\alpha T_p}{N_0} |\langle \boldsymbol{x}, \boldsymbol{s}_{\tau, f_o} \rangle| \right) \end{aligned} \tag{3.2}$$

where I_0 is the *modified Bessel function of first kind*.

The next step would be marginalization of the resulting LF over the frequency offset f_o.

$$p(\boldsymbol{x}|\boldsymbol{s},\tau,\alpha) = \int_{-\infty}^{\infty} p(\boldsymbol{x}|\boldsymbol{s},\tau,f_o,\alpha)p(f_o|\boldsymbol{s},\tau,\alpha)\mathrm{d}f_o \tag{3.3}$$

However, this step is hard to compute analytically due to the location of f_o in the Bessel function and unknown PDF of the frequency offset. The approximation of the marginalized LF could be computed by assuming f_o to be uniformly distributed on a given frequency interval. The mean value could then be approximated numerically as an average of the LF over a set of given frequency offsets.

However, this approach is computationally even more difficult than minimizing Equation 3.2 directly (with coarse f_o estimation), because in the latter method we can take advantage of monotonicity of increasing I_0, and minimize its argument only.

$$\hat{\tau}, \hat{f}_o = \arg\max_{\tau,f_o} |\langle \boldsymbol{x}, \boldsymbol{s}_{\tau,f_o}\rangle| \tag{3.4}$$

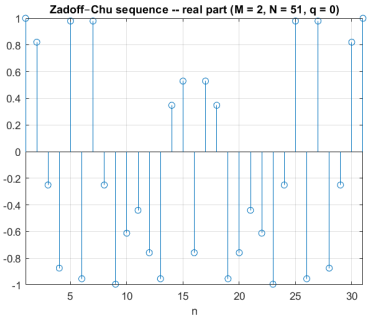
The distance between f_o should be chosen so that the resulting uncompensated offset would not much affect the ability of the delay estimator. It is good to choose it so that the sequence could not be rotated more than 90°.

Zadoff–Chu sequences

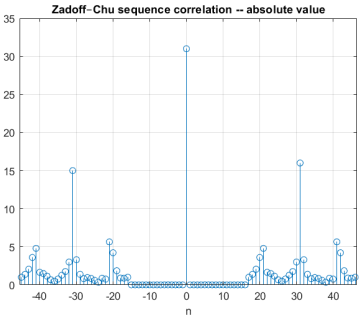
For the delay estimation we use the *Constant Amplitude Zero Autocorrelation* sequences (*CAZAC*), specifically the *Zadoff–Chu sequences*. Its property of zero cyclic autocorrelation function (except for the multiples of its period) gives us a good way of precisely determining the signal delay by searching for a significant peak in the cross-correlation function with the received signal. As the cross-correlation function is not cyclic, at least two periods of the sequence are used in the transmitted signal (one main sequence surrounded by two halves); however, other smaller peaks will still occur in it (see Figure 3.1).

For M and N coprime numbers and q an even number, Zadoff–Chu sequences with period N are defined as

$$s_n = \begin{cases} \exp(\mathrm{j}\pi\frac{M}{N}n(n+q)), & N \text{ even} \\ \exp(\mathrm{j}\pi\frac{M}{N}n(n+1+q)), & N \text{ odd} \end{cases} \tag{3.5}$$



(a) : Zadoff–Chu sequence



(b) : Zadoff–Chu sequence cross-correlation

Figure 3.1: Zadoff–Chu sequence and its cross-correlation

The peak autocorrelation property, however, implies constant spectral properties with wide bandwidth, so it is useful to modulate the sequence at least by a rectangular pulse (repeating each element). Ratio N_s is often chosen the same as in the following data modulation.

Pseudorandom-Noise sequences

Other sequences widely used in digital communications are the *Pseudorandom-Noise* (*PN*) sequences. They are binary, periodic sequences with a long period, easily generated by a shift register and a few XOR gates, mapped to elements $\{-1, 1\}$. They show autocorrelation properties similar to the Zadoff–Chu sequences. From their noise-like characteristic their autocorrelation ideally should be zero except for the multiplies of period. [1, p. 798]

Furthermore, their detection (and delay estimation) should be more resistant to an uncompensated frequency offset in the received signal as compared to the Zadoff–Chu sequences. (Their gradually increasing frequency is unpleasantly compatible with frequency offset, which then cannot be estimated in Equation 3.4 and spoils the delay estimation as well.)

Their generation can be described by a diagram like in Figure 3.2 or by a *generator polynomial* that determines the feedback lines.

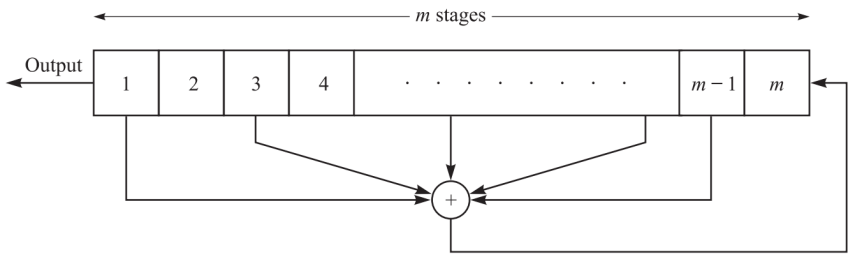


Figure 3.2: PN sequence generator[1, p. 797]

The goal is to generate a sequence with a maximum period preserving the

smallest length of the shift register. The sequences with maximized period are called *maximum-length* sequences, attaining length of $N = 2^m - 1$ bits generated by an m-stage shift register (this implies, that all states of the shift register, except zeros only, are part of the main cycle). Their autocorrelation function is then periodic and can have one of two values. [9, p. 63]

$$\mathcal{R}[k] = \begin{cases} 1, & k = Nl, l \in \mathbb{Z} \\ -\frac{1}{N}, & \text{otherwise} \end{cases} \tag{3.6}$$

As in Zadoff–Chu sequences, wide bandwidth of PN sequences is limited by modulation. It is appropriate to directly use the BPSK modulation.

3.1.2 Frequency offset estimation

For the finer frequency offset f_o estimation we return to Equation 3.4. Knowing the delay estimate $\hat{\tau}$ we can substitute it, and from Equation 2.29 we derive

$$\begin{aligned} \hat{f}_o &= \arg\max_{f_o} |\langle \boldsymbol{x}, \boldsymbol{s}_{\hat{\tau},f_o} \rangle| = \arg\max_{f_o} |\sum_k x[k] s^*_{\hat{\tau}}[k] e^{-\mathrm{j}2\pi f_o T_p k}| \\ &= \arg\max_{f_o} |\mathrm{DtFT}\left[x[k] s^*_{\hat{\tau}}[k]\right](f_o T_p)| \end{aligned} \tag{3.7}$$

As samples of the discrete Fourier transform (DtFT) are computed by the FFT algorithm we can quickly compute coarse frequency offset

$$\hat{f}_o T_p \approx \arg\max |\mathrm{FFT}_{\mathrm{shifted}}\left[x[k] s^*_{\hat{\tau}}[k]\right]| \tag{3.8}$$

For better accuracy, the DtFT samples can be analytically interpolated (around the maximum from FFT of length N) using the following formula. [8]

$$X(f) = \frac{1 - e^{-\mathrm{j}2\pi f}}{N} \sum_{m=0}^{N-1} \frac{\mathrm{FFT}\left[x[k] s^*_{\hat{\tau}}[k]\right](m)}{1 - e^{-\mathrm{j}2\pi (f-m)/N}} \tag{3.9}$$

For a precision estimate (how many DtFT samples should be computed) we will later develop the Cramér–Rao lower bound for f_o (in section 3.3).

For a good approximate of the frequency offset we use a harmonic signal whose Fourier transform forms a significant peak at its frequency (analogically to the Zadoff–Chu sequence). It is convenient to use a constant signal, since its frequency peak lies in zero and it is easy to generate.

$$s_{\hat{\tau}}[k] = 1 \tag{3.10}$$

3.1.3 Amplitude attenuation estimation

For the used frequency pilot signal the attenuation of amplitude can be easily estimated from the modulus of the frequency peak in the computed DtFT. This value expresses the energy of the signal which is not changed by frequency offset and phase, it can be computed as

$$|\mathrm{DtFT}\left[x[k]\right](\hat{f}_o T_p)| = |\left\langle \boldsymbol{x}, \boldsymbol{s}_{\hat{\tau},\hat{f}_o} \right\rangle| \approx |\left\langle \alpha e^{\mathrm{j}\varphi} \boldsymbol{s} + \boldsymbol{w}, \boldsymbol{s} \right\rangle| \approx \alpha \mathcal{E}_{\boldsymbol{s}} \tag{3.11}$$

$$\hat{\alpha} = \frac{|\mathrm{DtFT}\left[x[k]\right](\hat{f}_o T_p)|}{\mathcal{E}_{\boldsymbol{s}}} \tag{3.12}$$

Similarly $\hat{\alpha}$ could be computed even from the delay estimation sequence, but the precision would be better from the former sequence since the delay estimate is identical, and the frequency estimate is more precise.

3.1.4 Phase estimation

The phase φ is harder to estimate, since even a relatively low uncompensated frequency offset gradually changes the phase's value; moreover, the presence of phase noise and slight changes in frequency of oscillators in the transmitter and receiver makes the estimate even worse.

Especially for longer data transmissions, it is necessary to continually update the phase estimate. It can be done either by repeating a pilot signal, or by continuously tracking the phase directly on the modulated signal in constellation space.

For the QPSK modulated data signal, the tracking of the phase can be achieved by marginalization of the LF from Equation 2.34 over QPSK data (constellation symbols). Than the phase estimate can be written as

$$\hat{\varphi}[n] = \frac{1}{4}\arg\left(\sum_{k=n-W+1}^{n} x_{g,k,\hat{f}_o}^4\right) - \frac{\pi}{4} + m\frac{\pi}{2}, m \in \mathbb{Z} \tag{3.13}$$

where W denotes the length of the sliding window. The formula's derivation can be found in [4].

Since this method can determine the phase without knowledge of its shift $m\frac{\pi}{2}$, the initial value has to be known, and later values can then be tracked from its property of continuity. The initial value of the phase can be determined similarly as the attenuation of the amplitude from the phase of the frequency peak in the computed DtFT of the frequency pilot signal.

$$\hat{\varphi_0} = \arg\left(\mathrm{DtFT}\left[x[k]\right](\hat{f}_o T_p)\right) \tag{3.14}$$

3.2 Signal detection

For detection of the signal presence in the frame we will use the Neyman-Pearson theorem from subsection 2.5.2.

As the estimator determines the delay estimate (with coarse frequency estimation) from the maximum of $|\langle \boldsymbol{x}, \boldsymbol{s}_{\tau,f_o}\rangle|$ and gives its maximum $\sqrt{y} = |\left\langle \boldsymbol{x}, \boldsymbol{s}_{\hat{\tau},\hat{f}_o}\right\rangle|$, we can use it as an indicator bearing the information about the signal presence in the Neyman-Pearson theorem (for easier derivation we take its squared value).

At first, we define the hypotheses, and accordingly their assumed signals $\boldsymbol{s}^{(0)}$ and $\boldsymbol{s}^{(1)}$ (synchronized in frequency and time)

$$\begin{cases} \mathcal{H}_0 \dots y = |\left\langle \boldsymbol{s}^{(0)}_{\hat{\tau},\hat{f}_o}, \boldsymbol{s}_{\hat{\tau},\hat{f}_o} \right\rangle|^2 = |\left\langle \boldsymbol{s}^{(0)}, \boldsymbol{s} \right\rangle|^2 = |\langle \boldsymbol{0} + \boldsymbol{w}, \boldsymbol{s} \rangle|^2 \\ \mathcal{H}_1 \dots y = |\left\langle \boldsymbol{s}^{(1)}_{\hat{\tau},\hat{f}_o}, \boldsymbol{s}_{\hat{\tau},\hat{f}_o} \right\rangle|^2 = |\left\langle \boldsymbol{s}^{(1)}, \boldsymbol{s} \right\rangle|^2 = |\langle \alpha e^{\mathrm{j}\varphi} \boldsymbol{s} + \boldsymbol{w}, \boldsymbol{s} \rangle|^2 \end{cases} \tag{3.15}$$

Now, we substitute

$$\xi = \langle \boldsymbol{w}, \boldsymbol{s} \rangle \tag{3.16}$$

$$\begin{cases} \mathcal{H}_0 \dots y = |\xi|^2 \\ \mathcal{H}_1 \dots y = |\langle \alpha e^{\mathrm{j}\varphi} \boldsymbol{s}, \boldsymbol{s} \rangle + \xi|^2 = |\alpha e^{\mathrm{j}\varphi} \mathcal{E}_{\boldsymbol{s}} + \xi|^2 \end{cases} \tag{3.17}$$

In the next step we define normalized ξ' with the variance of two, so that we could later describe normalized y' by the *Noncentral Chi-Square* PDF (described in [1, p. 46]).

$$\xi' = \frac{\sqrt{2}}{\sigma_\xi} \xi \tag{3.18}$$

where

$$\sigma_\xi^2 = \mathrm{E}\left[|\xi|^2\right] = \mathrm{E}\left[\sum_k \sum_{k'} w_k s_k^* w_{k'}^* s_{k'}\right] = \sigma_{w_k}^2 \mathcal{E}_{\boldsymbol{s}} \tag{3.19}$$

Then we can write

$$\begin{cases} \mathcal{H}_0 \dots y' = \frac{2y}{\sigma_\xi^2} = |\frac{\sqrt{2}}{\sigma_\xi}\xi|^2 = |\xi'|^2 \\ \mathcal{H}_1 \dots y' = \frac{2y}{\sigma_\xi^2} = |\frac{\sqrt{2}}{\sigma_\xi}\alpha e^{\mathrm{j}\varphi}\mathcal{E}_{\boldsymbol{s}} + \frac{\sqrt{2}}{\sigma_\xi}\xi|^2 = |\frac{\sqrt{2}}{\sigma_\xi}\alpha e^{\mathrm{j}\varphi}\mathcal{E}_{\boldsymbol{s}} + \xi'|^2 \end{cases} \tag{3.20}$$

Subsequently, the Noncentral Chi-Square PDF of y' can be expressed as

$$p(y'|\boldsymbol{s}^0) = \frac{1}{2} e^{-\frac{y'}{2}} U(y') \tag{3.21}$$

$$p(y'|\boldsymbol{s}^1) = \frac{1}{2} e^{-\frac{y'+\lambda'}{2}} \mathrm{I}_0(\sqrt{\lambda' y'}) U(y') \tag{3.22}$$

where $U(y')$ denotes the *unit step* function, and for λ' we can write

$$\lambda' = |\frac{\sqrt{2}}{\sigma_\xi}\alpha e^{\mathrm{j}\varphi}\mathcal{E}_{\boldsymbol{s}}|^2 + 0^2 = \frac{2\alpha^2\mathcal{E}_{\boldsymbol{s}}^2}{\sigma_\xi^2} = \frac{2\alpha^2\mathcal{E}_{\boldsymbol{s}}}{\sigma_{w_k}^2} \tag{3.23}$$

Then the likelihood ratio is

$$\Lambda(y') = \frac{p(y'|\boldsymbol{s}^1)}{p(y'|\boldsymbol{s}^0)} = e^{-\frac{\lambda'}{2}} \mathrm{I}_0(\sqrt{\lambda' y'}) U(y') \tag{3.24}$$

and the Neyman-Pearson theorem states that the hypothesis $\mathcal{H}_1$ (the signal is present) should be chosen if and only if $\Lambda(y') > \gamma$

$$\mathcal{H}_1 \dots \Lambda(y') > \gamma \tag{3.25}$$

It is convenient to utilize monotonicity of increasing I_0 and find a constant β so that

$$\mathcal{H}_1 \ldots y' > \beta \tag{3.26}$$

This information tells us that the detector should detect the signal if and only if the y' (and so y) cross a lower boundary of detection.

The β can then be computed

$$P_{FA} = \int_{\beta}^{\infty} p(y'|\boldsymbol{s}^0)\mathrm{d}y' = \int_{\beta}^{\infty} \frac{1}{2} e^{-\frac{y'}{2}} U(y')\mathrm{d}y' = e^{-\frac{\beta}{2}} \tag{3.27}$$

$$\beta = -2\ln\left(P_{FA}\right) \tag{3.28}$$

Thus for the unnormalized y

$$\mathcal{H}_1 \ldots y > \frac{\beta\sigma_\xi^2}{2} = -\ln\left(P_{FA}\right)\sigma_{w_k}^2 \mathcal{E}_{\boldsymbol{s}} \tag{3.29}$$

To conclude, the detector should detect the signal if and only if the $y = |\left\langle \boldsymbol{x}, \boldsymbol{s}_{\hat{\tau},\hat{f}_o} \right\rangle|^2$ from the parameters estimator is greater than a constant computed according to Equation 3.29.

This detector, however, needs a high value of false alarm probability P_{FA} to reliably detect the sequence. The probability of detection can be computed as

$$P_{\mathrm{D}} = \int_{\beta}^{\infty} p(y'|\boldsymbol{s}^{(1)})\mathrm{d}y' \tag{3.30}$$

For used both Zadoff–Chu and PN sequences this problem can be mitigated by utilizing its zero cyclic autocorrelation values around the maximum. This is done by setting the P_{FA} relatively high (for example 0.1) with filtering false alarms by controlling low values of y near the maximum by an upper boundary.

3.3 CRLB of the frequency offset

As we know DtFT samples of the pilot signal for frequency offset (subsection 3.1.2), but do not know the precision to which we should interpolate it, we compute the maximal theoretical precision.

For the Cramér–Rao lower bound of the frequency offset we need to get rid of all other parameters than f_o in the LF. As already done before, the phase φ can be marginalized (Equation 3.2). Marginalization of other parameters, however, is analytically either unrealizable or difficult for computation. So as the delay and amplitude estimation is already done before, it can be used as approximation (the variance should then work out to be smaller than it actually is).

$$p(\boldsymbol{x}|\boldsymbol{s}, \hat{\tau}, f_o, \hat{\alpha}) = c_2(\hat{\alpha})\mathrm{I}_0\left(\frac{\hat{\alpha}T_p}{N_0}|\langle \boldsymbol{x}, \boldsymbol{s}_{\hat{\tau},f_o}\rangle|\right) \tag{3.31}$$

The next step is getting the derivative of the logarithm of the LF.

$$\ln p(\boldsymbol{x}|\boldsymbol{s}, \hat{\tau}, f_o, \hat{\alpha}) = \ln c_2(\hat{\alpha}) + \ln \mathrm{I}_0\left(\frac{\hat{\alpha}T_p}{N_0}|\langle \boldsymbol{x}, \boldsymbol{s}_{\hat{\tau},f_o}\rangle|\right) \tag{3.32}$$

$$\frac{\partial \ln p(\boldsymbol{x}|\boldsymbol{s},\hat{\tau},f_o,\hat{\alpha})}{\partial f_o} = \frac{\mathrm{I}_1\left(\frac{\hat{\alpha}T_p}{N_0}|\langle \boldsymbol{x},\boldsymbol{s}_{\hat{\tau},f_o}\rangle|\right)}{\mathrm{I}_0\left(\frac{\hat{\alpha}T_p}{N_0}|\langle \boldsymbol{x},\boldsymbol{s}_{\hat{\tau},f_o}\rangle|\right)}\frac{\hat{\alpha}T_p}{N_0}\frac{\partial|\langle \boldsymbol{x},\boldsymbol{s}_{\hat{\tau},f_o}\rangle|}{\partial f_o} \tag{3.33}$$

where the last element can be computed as

$$\begin{aligned}\frac{\partial|\langle \boldsymbol{x},\boldsymbol{s}_{\hat{\tau},f_o}\rangle|}{\partial f_o} &= \frac{1}{2|\langle \boldsymbol{x},\boldsymbol{s}_{\hat{\tau},f_o}\rangle|}\frac{\partial}{\partial f_o}\left(\langle \boldsymbol{x},\boldsymbol{s}_{\hat{\tau},f_o}\rangle\langle \boldsymbol{s}_{\hat{\tau},f_o},\boldsymbol{x}\rangle\right)\\ &= \frac{1}{2|\langle \boldsymbol{x},\boldsymbol{s}_{\hat{\tau},f_o}\rangle|}\Big(\frac{\partial}{\partial f_o}\left(\langle \boldsymbol{x},\boldsymbol{s}_{\hat{\tau},f_o}\rangle\right)\langle \boldsymbol{s}_{\hat{\tau},f_o},\boldsymbol{x}\rangle\\ &+ \langle \boldsymbol{x},\boldsymbol{s}_{\hat{\tau},f_o}\rangle\frac{\partial}{\partial f_o}\left(\langle \boldsymbol{s}_{\hat{\tau},f_o},\boldsymbol{x}\rangle\right)\Big)\\ &= \frac{1}{|\langle \boldsymbol{x},\boldsymbol{s}_{\hat{\tau},f_o}\rangle|}\Re\left[\langle \boldsymbol{x},\boldsymbol{s}_{\hat{\tau},f_o}\rangle^*\frac{\partial}{\partial f_o}\langle \boldsymbol{x},\boldsymbol{s}_{\hat{\tau},f_o}\rangle\right]\end{aligned} \tag{3.34}$$

Now we can substitute the particular pilot signal for which we use the constant signal $s_{\hat{\tau}}[k] = 1$ ($s_{\hat{\tau},f_o}[k] = 1e^{j2\pi f_o T_p k}$).

$$\langle \boldsymbol{x},\boldsymbol{s}_{\hat{\tau},f_o}\rangle = \sum_k x[k]e^{-j2\pi f_o T_p k} \tag{3.35}$$

$$\frac{\partial}{\partial f_o}\langle \boldsymbol{x},\boldsymbol{s}_{\hat{\tau},f_o}\rangle = -\mathrm{j}2\pi T_p\sum_k kx[k]e^{-j2\pi f_o T_p k} \tag{3.36}$$

The variance of estimate $\hat{f}_o$ can then be computed for a given set of predicted $\hat{f}_o$ and $\hat{\alpha}$ by Equation 2.38, The expectation operation in it will have to be computed numerically by random generation of φ and $\boldsymbol{w}$ according to

$$\boldsymbol{x} = \hat{\alpha}e^{\mathrm{j}\varphi}\boldsymbol{s}_{\hat{\tau},\hat{f}_o} + \boldsymbol{w} \tag{3.37}$$

Chapter 4

Implemented models

4.1 System blocks implementation

The digital communication system can be implemented block-wise from blocks outlined in section 2.1. These blocks will be used for construction of the simulation model in MATLAB and subsequently for the real signal transmission. Parameters for the blocks and MATLAB simulator will be set to resemble those in the SDR (see subsection 4.3.1).

4.1.1 Convolutional code

The convolutional coder (described in subsection 2.2.1) is implemented in function `conv_code`. It creates coded data `c` from input data `b` with the constraint length of 2 and code rate 1/2.

Used generator matrices $\boldsymbol{G}_i$ are three changeable matrices; in the code they are merged to the variable `G`.

$$\boldsymbol{G}_0 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \boldsymbol{G}_1 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \boldsymbol{G}_2 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \tag{4.1}$$

Computation of the coded data `c` is done conveniently by convolution and modulo operation of the input data `b` and reversed rows of `G`.

```
1 c = zeros(2, parameters.b_number+2); %conv adds +2 elements
2
3 c(1,:) = conv(b, parameters.G(1,[3,2,1]));
4 c(2,:) = conv(b, parameters.G(2,[3,2,1]));
5
6 c(:,[parameters.b_number+1,parameters.b_number+2]) = [];
7 c = mod(c, 2);
```

Variable `b_number` denotes the length of the data `b` sequence.

4.1.2 QPSK modulation

Coded data `c` is modulated by the QPSK modulation in function `modulation` into the modulated signal `s`. Ratio `Ns` between symbol period `Ts` and sampling

period Tp is selected Ns = 4.

$$\mathtt{Ts} = \mathtt{Ns} \cdot \mathtt{Tp} = 4\mathtt{Tp} \tag{4.2}$$

Codewords c are converted into (complex) channel symbols q, as described in subsection 2.3.1. c(:,n) and q(n) elements are mapped directly because both can have one of four values. s is then computed as convolution of q upsampled by Ns and pulse g according to the method presented in subsection 2.3.3 and scaled by appropriate modulation_Gain = 143.57, so that the largest possible sample would fit into the *int16* datatype (computed in modulation_gain.m). Described modulation process is shown below.

```
1 q = zeros(1, parameters.b_number);
2 for n = 1:parameters.b_number
3     q(n) = 1/sqrt(2)*(-1 +2*c(1, n) + 1i*(-1+2*c(2, n)));
4 end
5
6 q_upsampled = zeros(1, ...
      parameters.Ns*parameters.b_number-parameters.Ns+1);
7 q_upsampled(1:parameters.Ns:parameters.Ns * ...
      parameters.b_number) = q; %upsampling
8 s = conv(q_upsampled, g);
9 s = s*parameters.modulation_Gain;
```

As symbol pulse g, the RRC pulse sampled around zero time is used. Parameters of the pulse are alpha = 0.5 (roll-off factor) and L = 10, where L · Ns gives the number of samples of the pulse. RRC pulse is generated in function RRC.

4.1.3 Pilot signals

The generation of pilot signals for the parameters estimation is implemented in function pilot_signal. The type of pilot signal for delay synchronization (Zadoff–Chu or PN sequence) can be chosen in input variable parameters.pilot.type. The second pilot signal for frequency synchronization is constant (for simplicity 1) with a length of 1024. Both signals are normalized to the maximum of *int16* datatype by Gain = 32767.

Zadoff–Chu sequence

The Zadoff–Chu pilot signal is generated with parameters $M = 3$, $N = 31$, $q = 0$. Two periods (one main period surrounded by two halves) of the sequence modulated by rectangular pulse of length rec_factor = 4 are used, giving the total length of 248. (see subsection 3.1.1)

PN sequence

The second signal used for delay synchronization is a maximum-length PN sequence with a 5-stage shift register and length of 31 (chosen intentionally

with the same length as the Zadoff–Chu sequence). Its generator polynomial coefficients are $[1, 0, 0, 1, 0, 1]$. Two periods modulated to the total length of 248 by rectangular pulse are used as well.

4.1.4 AWGN channel simulation

The simulator of AWGN channel with constant signal delay `tau`, amplitude attenuation `alpha`, frequency offset `freqoffset`, and phase `phi` is done straightforwardly in function `AWGN_channel`.

The function takes the original signal `s_complete`, multiplies it with `alpha` and complex exponentials, one with `phi` and the second element-wise with `freqoffset`. Then it places this sequence into a frame of length `Frame`, adds AWGN noise to the whole frame and stores the output into the variable `x`.

```
%add alpha, phi
x_notdelayed = s_complete * channel_parameters.alpha ...
    *exp(1i*channel_parameters.phi);

%add freq shift
k = 0:parameters.N_s_complete-1;
x_notdelayed = x_notdelayed.*exp(1i ...
    *2*pi*channel_parameters.freqoffset*k);

%add delay tau
x = zeros(1,channel_parameters.Frame);
x(channel_parameters.tau+1: ...
    channel_parameters.tau+parameters.N_s_complete) = ...
    x_notdelayed;

%add noise
w = sqrt(2*channel_parameters.N0/parameters.Tp)* (randn([1, ...
    channel_parameters.Frame])+ 1i*randn([1, ...
    channel_parameters.Frame]))/sqrt(2);
x = x + w; %add w
```

4.1.5 Parameters estimation and signal detection

The estimation of unknown channel parameters is done in two different ways with two pilot signals (Zadoff–Chu and PN sequence) since the delay estimation without known frequency offset is the key step to the whole data reception. At the same time there are problems either with computational complexity (when we estimate delay and frequency offset at the same time) or with delay estimation accuracy (when we ignore the frequency offset).

Joint estimator for delay and frequency offset

Estimating both delay and frequency offset as derived in Equation 3.4 can be done with both Zadoff–Chu and PN sequences. This approach is implemented in function `estimation_joint`.

As stated in subsection 3.1.1, the sequence should not be rotated more than 90° in the duration of estimation, so the step in tested frequency offsets should be chosen sufficiently small (however smaller steps increase the computational time).

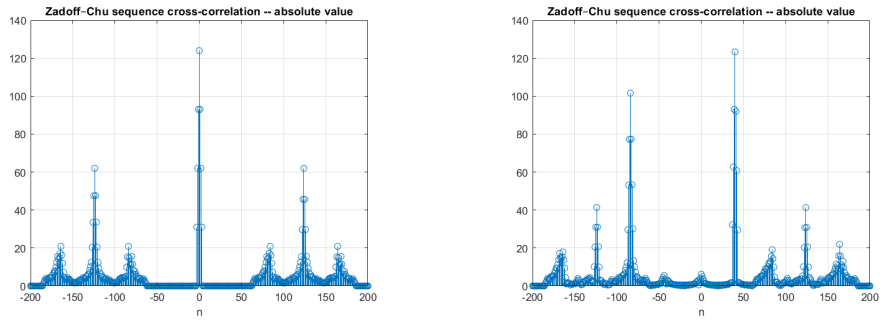
$$\frac{\pi}{2} \geq \Delta\varphi = 2\pi\Delta f_o T_p L \tag{4.3}$$

L in Equation 4.3 denotes the number of pilot signal samples in one (searched) period. $L = 124$. Thus

$$\Delta f_o \leq \frac{f_p}{4L} \approx 394 \text{ Hz} \tag{4.4}$$

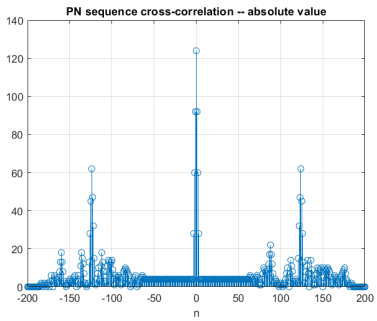
Since the estimation should lie at most 394 Hz from a tested frequency offset, the step in tested frequency offsets can be two times greater. We choose the frequency step `f_step` = 750 Hz with an adequate maximum offset of `max_f_offset` = 3750 Hz.

In this approach, however, a problem occurred with the Zadoff–Chu sequence whose LF (cross-correlation) is nearly constant with frequency offset (thanks to its inner structure of linearly increasing frequency). Then the frequency offset could not be estimated, and without this knowledge there occurs delay estimation error. On the other hand, the PN sequence is not based on changing frequency and its LF decreases with present frequency offset.

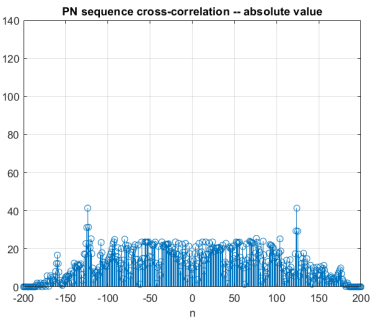


(a) : frequency offset 0 Hz **(b) :** frequency offset 1500 Hz

Figure 4.1: Cross-correlation of used modulated Zadoff–Chu sequence



(a) : frequency offset 0 Hz



(b) : frequency offset 1500 Hz

Figure 4.2: Cross-correlation of used modulated Pseudorandom-Noise sequence

Iterative estimator of delay and frequency offset

The unpleasant property of constant LF in the Zadoff–Chu sequence over frequency offset can be simply utilized. If the LF will have a maximum for any frequency offset, then the delay can be coarsely estimated. With a coarse delay estimate, the section of received signal containing the frequency offset pilot signal can be found. Knowing the frequency offset, the delay estimate can be easily corrected.

This approach can be found in function `estimation_iterative`. It does not work for the PN sequence since its LF maximum diminishes with the present frequency offset. It is significantly faster than the joint estimator described above with possible usage of longer sequences.

Signal detection

The signal detection (described in subsection 2.5.2) should be realized from the Neyman-Pearson theorem. It is implemented in script `NeymanPearson.m`, where the lower boundary β and normalized correlation y' for noise signals are computed. For this, generated noise with normal distribution, then the same rounded noise, and lastly the received noise with the same variance are used.

The outcomes went not as expected for the received noise. For the given false alarm probability $P_{\mathrm{FA}} = 0.1$ the computed false alarm checking gives the true false alarm probability of $P_{\mathrm{FA}} \approx 0.4$. The suspicion fell on the discrete received values, but this was mitigated as the generated noise with normal distribution and its discrete version (obtained by rounding) give the expected values of the false alarm probability.

The solution was found by setting up the lower and upper boundaries (of the unnormalized y) reasonably from received values to `threshold_up` $= 3\ 10^7$ and `threshold_low` $= 2\ 10^7$.

CRLB for frequency offset estimation

For the possible precision of the frequency offset estimation we compute the CRLB (according to the method presented in section 3.3) using values covering the predicted range of values of the frequency offset and the amplitude attenuation (see subsection 4.3.1).

The computation of CRLB standard deviations of f_o is implemented in MATLAB script `CRLB.m`. It gives following values

$\mathrm{var}\left[\hat{\theta}\right]$	$f_o = 0$	$f_o = 500$	$f_o = 1000$	$f_o = 2000$	$f_o = 3000$	$f_o = 4000$
$\alpha = 10^{-4}$	1.4182	1.4212	1.4442	1.4038	1.4257	1.4189
$\alpha = 10^{-3}$	0.1415	0.1416	0.1397	0.1413	0.1428	0.1416
$\alpha = 10^{-2}$	0.0143	0.0141	0.0141	0.0142	0.0142	0.0143
$\alpha = 10^{-1}$	0.0014	0.0014	0.0014	0.0014	0.0014	0.0014
$\alpha = 10^{0}$	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001

Table 4.1: Computed CRLB of f_o standard deviations

From the table above we can deduce that the frequency offset estimate should be realizable and its results will ideally have extremely small errors of units of hertzs. (However, the standard deviations will be greater because of neglected phase noise.)

Frequency offset estimation

The frequency offset estimation is equally implemented in functions `estimation_joint` and `estimation_iterative`.

According to subsection 3.1.2, the FFT of the received frequency offset pilot signal is computed first, then more precise DtFT is computed around the FFT maximum frequency.

The FFT precision alone for the length of the pilot signal of 1024 and given f_p is

$$\Delta f_o = \frac{1}{1024} f_p = 190.7349 \text{ Hz} \tag{4.5}$$

The DtFT interpolation around the FFT maximum frequency is done symmetrical by $N = 2^n$ points. The number of DtFT points interpolated from FFT was chosen to be $N = 32$, this yields the frequency offset precision of

$$\Delta f_o' = \frac{\Delta f_o}{N} \approx 6 \text{ Hz} \tag{4.6}$$

which should be attainable considering the CRLB computed above.

4.1.6 Signal decomposition and equalization

Received signal decomposition `x_g` into the basis of shifted modulation pulse `g` is computed in function `g_decomposition`.

The signal is first equalized by the estimated delay `tau`, amplitude attenuation `alpha` (then the `x_g` can be directly compared in amplitude to channel symbols `q` and the maximum position of the LLF does not change), and frequency offset `freqoffset` with another phase component accumulated during the duration of the pilot signal for frequency offset estimation.

Then the signal is decomposed according to Equation 2.42.

Lastly, the phase of the signal is equalized in the constellation space by the method for QPSK phase synchronization described in subsection 3.1.4.

Such an equalized decomposed signal can be directly compared to the channel symbols `q` and detected according to the QPSK decision regions in Figure 2.3. This approach, however, does not consider prior data coding, so it can be later used for comparison of error rates between the coded and uncoded data.

4.1.7 Viterbi decoder

The sent data `b` from the equalized received signal samples in constellation space `x_g` is acquired by function `decoding_Viterbi`. This function decodes the data according to the Viterbi algorithm described in subsection 2.6.1. Generator matrices of the convolution code $\boldsymbol{G}_i$ can be seen in subsection 4.1.1.

From the knowledge of $\boldsymbol{G}_i$ we can define the modulator states as

$$\sigma_n^{(i)} = \begin{cases} \sigma_n^{(0)}, & b_{n-2} = 0, b_{n-1} = 0 \\ \sigma_n^{(1)}, & b_{n-2} = 0, b_{n-1} = 1 \\ \sigma_n^{(2)}, & b_{n-2} = 1, b_{n-1} = 0 \\ \sigma_n^{(3)}, & b_{n-2} = 1, b_{n-1} = 1 \end{cases} \tag{4.7}$$

(starting from one in the MATLAB code), and make trellis of the used convolutional code with transitions depending on the particular origin modulator state σ_n and actual data symbol b_n. To each transition it is possible to compute the respective codeword $\boldsymbol{c}_n$ from Equation 2.2 and the QPSK channel symbol q_n from Equation 2.8 (shifted by $\frac{\pi}{4}$).

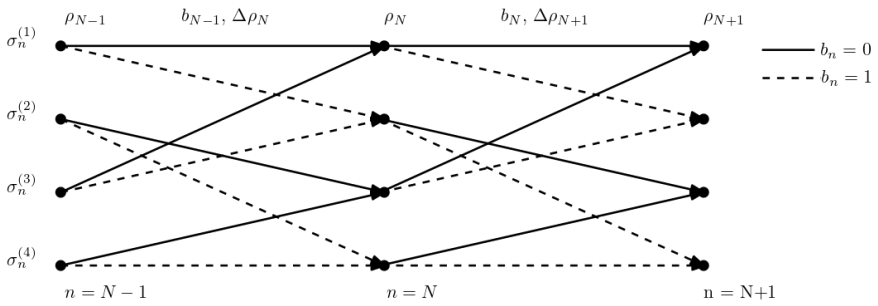


Figure 4.3: Transitions in modulator states

	$b_n = 0$	$b_n = 1$
$\sigma_n^{(0)}$	$\boldsymbol{c}_n = \begin{bmatrix}0\\0\end{bmatrix}$	$\boldsymbol{c}_n = \begin{bmatrix}1\\1\end{bmatrix}$
$\sigma_n^{(1)}$	$\boldsymbol{c}_n = \begin{bmatrix}0\\1\end{bmatrix}$	$\boldsymbol{c}_n = \begin{bmatrix}1\\0\end{bmatrix}$
$\sigma_n^{(2)}$	$\boldsymbol{c}_n = \begin{bmatrix}1\\1\end{bmatrix}$	$\boldsymbol{c}_n = \begin{bmatrix}0\\0\end{bmatrix}$
$\sigma_n^{(3)}$	$\boldsymbol{c}_n = \begin{bmatrix}1\\0\end{bmatrix}$	$\boldsymbol{c}_n = \begin{bmatrix}0\\1\end{bmatrix}$

(a) : Codeword of each transition

	$b_n = 0$	$b_n = 1$
$\sigma_n^{(0)}$	$q_n = \frac{-1-j}{\sqrt{2}}$	$q_n = \frac{1+j}{\sqrt{2}}$
$\sigma_n^{(1)}$	$q_n = \frac{-1+j}{\sqrt{2}}$	$q_n = \frac{1+j}{\sqrt{2}}$
$\sigma_n^{(2)}$	$q_n = \frac{1+j}{\sqrt{2}}$	$q_n = \frac{-1-j}{\sqrt{2}}$
$\sigma_n^{(3)}$	$q_n = \frac{1-j}{\sqrt{2}}$	$q_n = \frac{-1+j}{\sqrt{2}}$

(b) : QPSK channel symbol of each transition

Table 4.2: Codeword and QPSK channel symbol of each transition

The function thus first generates q_n for all $\sigma_n^{(i)}$ and b_n and stores them to the variable `channel_symbols`. Then it recursively computes all possible $\rho_{n+1}^{(i)}$ for every state $\sigma_{n+1}^{(i)}$ according to Equation 2.48 (with the difference that $\boldsymbol{x}_g$ has already been equalized by all parameters including amplitude) and for each state chooses the minimal one $\rho_{n+1}^{(i)}$ (`rho_new`) and saves its origin state into `paths`.

```
1  %Recurrent phase
2  for n = 1:parameters.b_number
3      for m = 1:8
4          delta_rho(m) = 1/2*(abs(channel_symbols{m} - ...
               x_g(n))).^2;
5      end
6      for m = 0:1 %b_{n-1}
7          for p = 0:1 %b_n
8              if ( rho_old(0+m +1) + delta_rho(0+m+1,p+1) <= ...
                   rho_old(2+m +1) + delta_rho(2+m+1,p+1) ) ...
                   %b_{n-2} = 0
9                  rho_new(2*m+p+1) = rho_old(0+m +1) + ...
                       delta_rho(0+m+1,p+1);
10                 paths(2*m+p+1,n+1) = 0+m+1; %came from 0+m+1 ...
                       sigma_n to this sigma_{n+1}
11             else %b_{n-2} = 1
12                 rho_new(2*m+p+1) = rho_old(2 +m +1) + ...
                       delta_rho(2+m+1,p+1);
13                 paths(2*m+p+1,n+1) = 2+m+1; %came from 2+m+1 ...
                       sigma_n to this sigma_{n+1}
14             end
15         end
16     end
17     rho_old = rho_new; %all 4 at once
18 end
```

This process starts at the defined state $\sigma_0 = \sigma_0^{(0)}$. This is done manually by setting all other $\rho_0^{(i)}$ at infinite values.

Then the optimal path is found backwards from the saved origin states. Because all transitions with $b_{n-1} = 1$ lead to states $\sigma_n^{(1)}$ and $\sigma_n^{(3)}$, it is very

easy to find $\hat{\boldsymbol{b}}$ (`b_result` in code) by tracking these states.

```
%Forward phase
b_result = zeros(1,parameters.b_number);%without 2 ending zeros
state = 1; %ending state (after flush sequence 00)
for n = parameters.b_number:-1:1
    if (state == 2 || state == 4)
        b_result(n) = 1;
    end
    state = paths(state,n+1);
end
b_result(parameters.b_number-1:parameters.b_number) = []; ...
    %without 2 ending zeros
```

4.2 MATLAB simulation

According to the overall system model in section 2.1, the simulator in MATLAB script `BC_MATLAB_simulation.m` consists of blocks of coder, QPSK modulator, pilot signal generator, AWGN channel model simulator, parameters estimator, signal basis decomposer and Viterbi decoder. All these blocks work on a per-frame principle; each one block processes its data as a complete frame.

Firstly, the script reads bit-wise data `b` from the saved file (image `logotx.jpg` of size 3.58 KB) and loads system parameters by function `parameters_load`. Then it codes the data by `conv_code` to `c` and modulates `c` by `modulation` to `s`. After this it merges `s` with the generated pilot signal `pil_s` to the complete signal `s_complete` ready for the channel simulation (or ready for the transmission by the SDR in the next chapter).

Then it simulates the channel by function `AWGN_channel` with parameters similar to those measured in the SDR (the variance of noise $\sigma_{w_k} = 400$ is taken larger for a better demonstration).

Then it estimates the channel parameters by function `estimation_joint` which works for the PN pilot signal or by `estimation_iterative` which works for the Zadoff-Chu pilot signal (see subsection 4.1.5). Then decomposition and decoding done by functions `g_decomposition` and `decoding_Viterbi` follow.

Lastly, the received data are saved (to the file: `logorx.jpg`), and graphs and the received image are rendered. This includes graph of the original, decoded data, and errors (with the computed error rate for this data) in Figure 4.4 (zoomed). Next graph in Figure 4.5 shows the used RRC pulse, modulated signal, and this signal after the AWGN channel simulation. The graph in Figure 4.6 shows delay estimation, Figure 4.7 then shows the double frequency offset estimation, and Figure 4.8 shows the phase estimation done by tracking this phase. The last graph in Figure 4.9 shows the received decomposed `x_g` in the constellation space with the computed error rate of detection (green points denote detection errors).

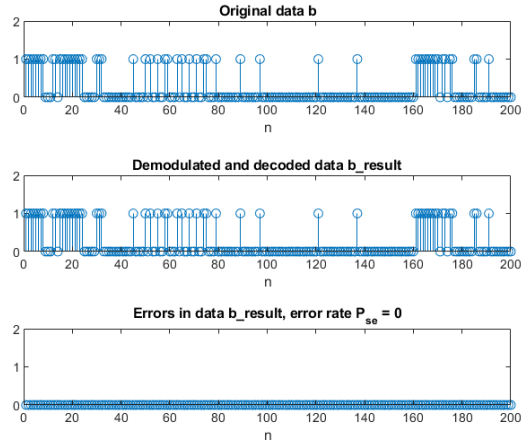


Figure 4.4: Comparison of sent and received data (zoomed)

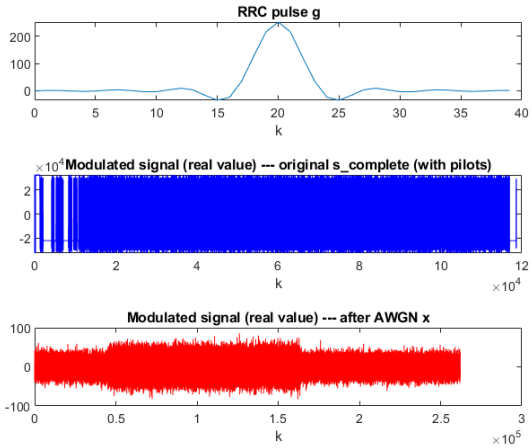


Figure 4.5: Used RRC pulse and modulated signal

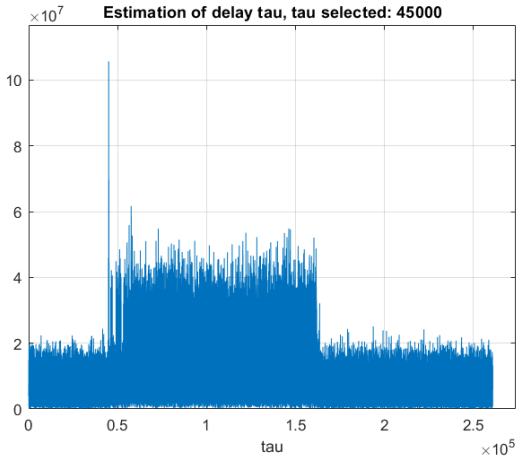


Figure 4.6: Estimation of delay

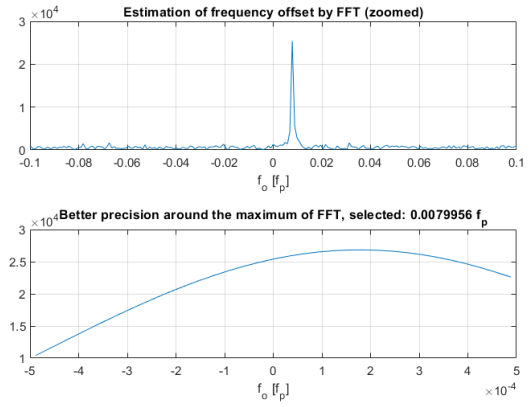


Figure 4.7: Estimation of frequency offset

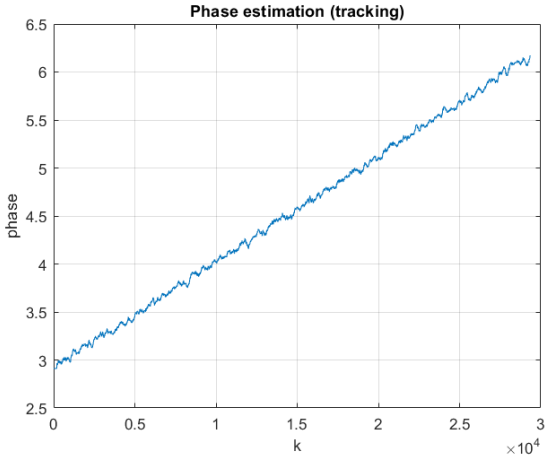


Figure 4.8: Phase estimation

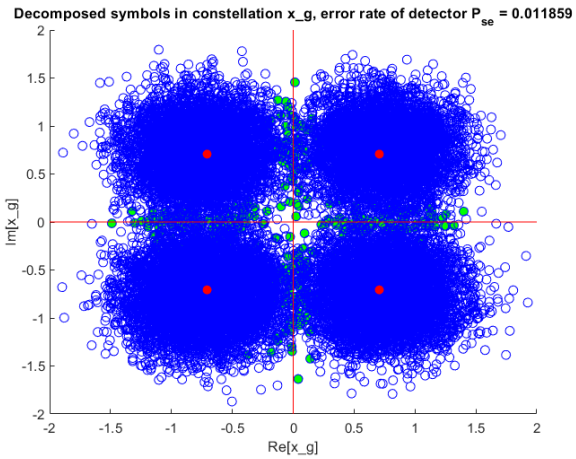


Figure 4.9: Decomposed received signal in constellation space

4.3 SDR implementation

In this section the channel model from the implemented simulator will be appropriately replaced with a radio transmitter and a receiver connected by a real over-the-air channel. Software defined radios (SDR) will be used for both the transmitter and the receiver.

4.3.1 Ettus USRP N210 SDR

SDRs used for the signal transmission are *USRP N210* from the company *Ettus Research™*. [5]

These radios can be used both as receivers and transmitters. In receiver mode they process the signal, sample it by 14-bit 100 MHz ADC, decimate it and convert resulting samples into the complex envelope. These samples are each sent as two numbers in *int16* datatype over the LAN network. In transmitter mode this process is reversed with the difference of usage of a 16-bit DAC.

For our purposes we used the *Internal* option for *PPSSource*, and *ClockSource* so that the radios' clocks would not be synchronized by a wire (as it is in real communication). *Gains* of radios are set experimentally to 0 at the transmitter and 16 at the receiver so that the signal would be reasonably strong against the noise when the antennas are placed close to each other, and similar to noise when they are farther apart. *Interpolation* and *Decimation factors* were set to 512, yielding the sampling frequency $f_s = 195.3125$ kHz. The *center frequency* was chosen at 2.45 GHz.

It was found out that the variance of received noise samples does not depend much on the *Gain* in the receiver; it was computed in `noise_var_f_o.m` for the used `Gain` of 16, that the variance is $\sigma_{w_k} \approx 4.1$. Simultaneously, it was found out that the frequency offset contains values of hundreds or thousands of Hz; in `noise_var_f_o.m` the value was computed at $f_o \approx 543$ Hz (from sent constant signal) with frequency peak in the power spectral density in Figure 4.10. This value, however, significantly fluctuates over time. The amplitude attenuation was measured $\alpha = 4.9\ 10^{-4}$.

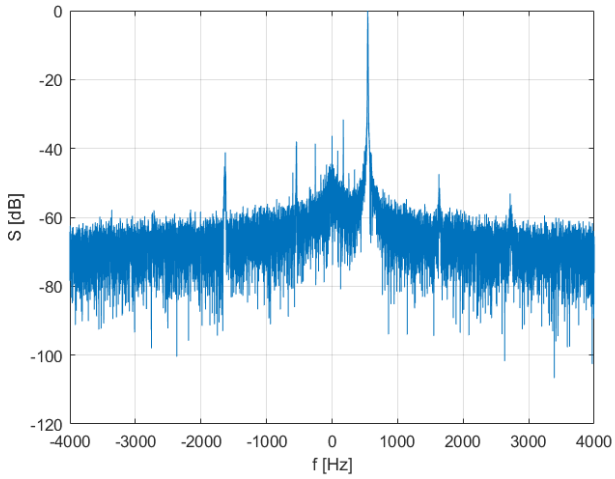


Figure 4.10: PSD with measured frequency offset

4.3.2 Data transmission using SDRs

The data transmission using SDRs differs mainly in separation of the code parts responsible for transmitting and receiving. Transmission is managed by the system object `comm.SDRuTransmitter` and reception by `comm.SDRuReceiver`.

There are implemented two models of data transmission using these objects.

Picture transmission

The picture transmission implemented in scripts `BC_pict_tx.m` and `BC_pict_rx.m` resembles the MATLAB simulation script. The structure of the transmitter is the same with a difference that the modulated data signal with pilot signals (`s_complete`) is passed to the transmitter object. It is done all without dividing of the data or the signal.

The receiver then reads two frames (`x` and `buffer`, slightly longer than the transmitted signal) and detects the delay pilot signal presence in the older frame `x` with the channel parameters estimation. If a pilot signal is present the located data signal is decomposed, decoded, saved, and graphs and the the received picture are rendered. As the pilot signal can be located everywhere in the frame, the data signal is separated from both `x` and `buffer`. Then the `buffer` is loaded into `x` and new signal is read to `buffer`. If there is no signal detected in `x` the data processing is skipped and new data are loaded. This process is endlessly repeated.

Figure 4.11 shows a performed delay estimation, we should point out the change of amplitude which can bring a problem for the set boundaries of detection. Figure 4.12 shows the phase estimation, the phase is clearly nonlinear with a visible carrier frequency change.

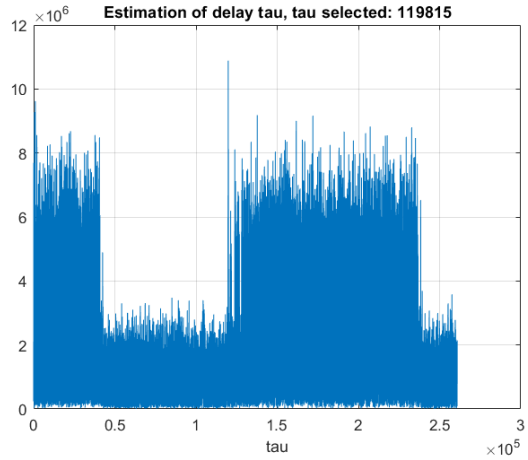


Figure 4.11: Estimation of delay (true)

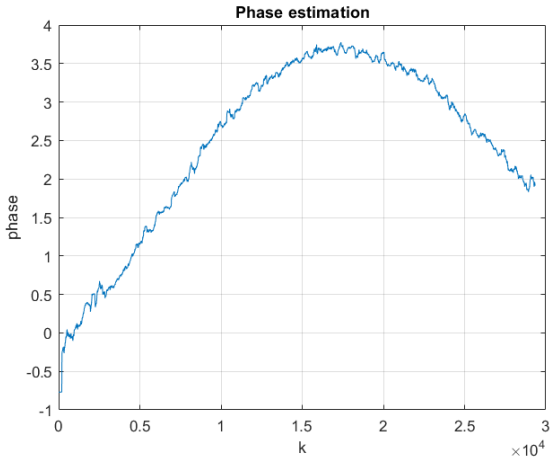


Figure 4.12: Phase estimation (true)

Arbitrary file transmission

The arbitrary file transmission implemented in scripts `BC_file_tx.m` and `BC_file_rx.m` is an attempt for a generalization of the dedicated picture transmission. It can transmit an arbitrary file long up to 16 MB with name long up to 20 characters.

The transmitter loads the file first and cut it into packets of length 1 KB (including the header), then it sends these packets one by one in the same way as the previous transmitter. All headers consist of the number of that actual packet (using two bytes), the zeroth packet's header also includes the length of the whole file in bytes (using three bytes) and the name of the file (using 20 bytes).

The receiver works similarly as the previous receiver with a difference that in the end it reads the header of the received packet and determines the following file operations. When the zeroth packet is received the number

of packets is computed and a file is generated. Name of the new file is the transmitted one with the prefix `rx1_` (or with a higher number in case of existence of such file name)

This process is not made flawlessly, corrupted headers and undetected packets are partially solved by writing zeros into the file when a packet with higher number arrives (at most 2 packets can be filled by zeros at a time). Moreover, it is not solved a problem with pilot signals lying in the end of packets, these pilot signals are ignored, this problem could be solved similarly as for data by reading the end of such pilot signal from the `buffer`.

An example of an output of this process can be seen in Figure 4.13, the sent bitmap image (`logotx.bmp`) is 199 KB long.

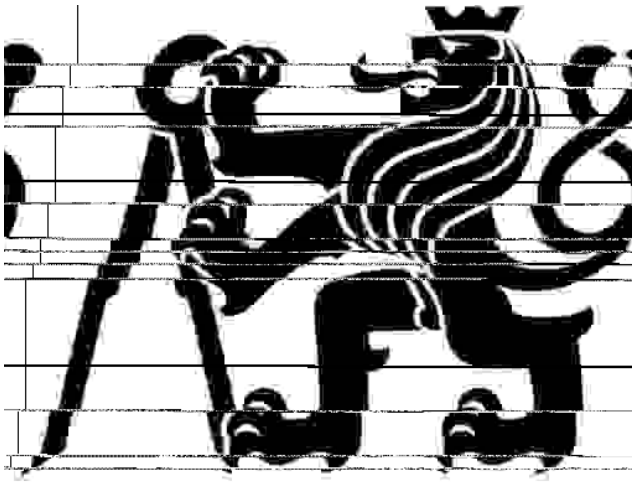


Figure 4.13: Received bitmap image

Chapter 5

Conclusion

In chapter 2 we got acquainted with the fundamental theory and basic principles of a simple digital communication. This included data coding, digital modulation, channel model description, signal synchronization and subsequent decoding as well as fundamental limits and conditions.

Next, in chapter 3 we used the theory to develop particular tools for the signal parameters estimation and signal detection.

Finally, in chapter 4 we used the prepared resources for the simulation of a digital communication in MATLAB. Then we used software defined radios for a true over-the-air communication.

In this project many things could be improved and development further. The algorithms and functions were implemented illustratively and they should be later optimized. Next, the exploited effect of an uncompensated frequency offset on Zadoff–Chu sequences (in subsection 4.1.5) should be mathematically proved. Another improvements could be done in script `BC_file_rx.m` for an arbitrary file transmission (see subsection 4.3.2).

To conclude the goal of basic transmission in the test bed was accomplished. Moreover, the scripts render various graphs so they can be used as visual examples of implemented methods.

The next work could be focused on an implementation of the verified methods in real-time low-level applications, for example, in a microcontroller.

Bibliography

[1] PROAKIS, John G. and Masoud SALEHI. *Digital Communications. 5.* New York: McGraw - Hill, 2008. ISBN 978–0–07–295716–7.

[2] SÝKORA, Jan. *Digital Communications.* – lecture slides. 2021.

[3] SÝKORA, Jan. *Statistical Signal Processing* – lecture slides. 2021.

[4] SÝKORA, Jan. *Synchronization and Equalization in Digital Communications* – lecture slides. 2021.

[5] Ettus Research, USRP N200/N210 datasheet. Available: `http://www.ettus.com/wp-content/uploads/2019/01/07495_Ettus_N200-210_DS_Flyer_HR_1.pdf`

[6] Wikimedia Commons contributors. *File:Root-raised-cosine-impulse.svg* [Internet]. Wikimedia Commons, the free media repository; 2020 Oct 28, 15:33 UTC [cited 2022 May 17]. Available: `https://commons.wikimedia.org/w/index.php?title=File:Root-raised-cosine-impulse.svg&oldid=505256403`.

[7] PATIL, Mrinmayi V., Sanjay PAWAR and Zia SAQUIB. *Coding Techniques for 5G Networks: A Review.* 2020 3rd International Conference on Communication System, Computing and IT Applications (CSCITA), 2020, pp. 208-213, doi: 10.1109/CSCITA47329.2020.9137797.

[8] LYONS, Rick. *FFT Interpolation Based on FFT Samples: A Detective Story With a Surprise Ending.* DSPRelated.com [online]. April 16, 2018 [cit. May 10, 2022]. Available: `https://www.dsprelated.com/showarticle/1156.php`

[9] KOVÁŘ, Pavel. *Družicová navigace: od teorie k aplikací v softwarovém přijímači.* Praha: České vysoké učení technické v Praze, Česká technika - nakladatelství ČVUT, 2016. ISBN 978-800-1059-890.

Appendix A

Attached files

```
/
├── computations
│   ├── CRLB.m
│   ├── modulation_gain.m
│   ├── NeymanPearson.m
│   ├── noise_var_f_o.m
│   ├── x_const_Gain16.txt
│   └── x_noise_Gain16.txt
├── figures
│   ├── BPSK_QPSK_dec_regions.m
│   ├── PN_f_o.m
│   ├── zadoffchu.m
│   └── zadoffchu_f_o.m
└── implementation
    ├── AWGN_channel.m
    ├── BC_file_rx.m
    ├── BC_file_tx.m
    ├── BC_matlab_simulation.m
    ├── BC_pict_rx.m
    ├── BC_pict_tx.m
    ├── conv_code.m
    ├── decoding_Viterbi.m
    ├── estimation_iterative.m
    ├── estimation_joint.m
    ├── g_decomposition.m
    ├── modulation.m
    ├── parameters_load.m
    ├── pilot_signal.m
    ├── RRC.m
    ├── logotx.bmp
    └── logotx.jpg
```