Die approbierte Originalversion dieser Diplom-/Masterarbeit ist an der Hauptbibliothek der Technischen Universität Wien aufgestellt (http://www.ub.tuwien.ac.at).

The approved original version of this diploma or master thesis is available at the main library of the Vienna University of Technology (http://www.ub.tuwien.ac.at/englweb/).

Diplomarbeit

Envelope Detector based on Hilbert Transform

ausgeführt am
Institut für Nachrichtentechnik und Hochfrequenztechnik
der Technischen Universität Wien

von

Florian Xaver
0225895
Wurzbachgasse 16, 1140 Purkersdorf

Wien, Juni 2009

Betreuer:

Dipl.-Ing. Ralph Prestros
(NXP Semiconductors Austria GmbH Styria)

Ass.Prof. Dipl.-Ing. Dr.techn. Gerhard Doblinger

Univ.Prof. Dipl.-Ing. Dr.-Ing. Christoph Mecklenbräuker

Abstract

This thesis deals with an envelope detector in a test case of ePassport readers. For the transmission between passports and readers RFID (Radio Frequency Identification) is being used. The standard of test methods for ePassport is published in [ISO] and contains a source code which applies an envelope detector based on HILBERT transform to a measured input signal. The envelope has to fulfil a standardized shape. This detector will be derived followed by an analysis of an implementation postulated in [ISO]. This will lead to a list of properties and possible improvements. Based on this description a test case will be discussed using a simulation developed in Matlab®. This testbed will contain a model of the signal source whose output signal is the input signal of a measurement path.

Keywords: Hilbert transform, complex/analytic signal, Gibbs phenomenon, windows, discretization error of floating-point variables, Type A signal, RFID, harmonics, non-linear system, Gaussian lowpass filter, cubic amplifier model, sampler, time jitter, quantizer, additive Gaussian noise, mean squared error

Zusammenfassung

Diese Diplomarbeit behandelt einen Hüllkurvendetektor zum Testen eines ePassport-Lesegeräts. Zur Datenübertragung zwischen einem Pass und einem dazugehörigen Lesegerät wird RFID (Radio Frequency Identification) verwendet. Damit es dem Standard [ISO] genügt, wird eine gewisse Hüllkurve des Empfangssignals gefordert. Zur Detektion dieser wird ein Hüllkurvendetektor, basierend auf der HILBERT-Transformation, vorgeschlagen. Im Folgenden wird dieser analysiert (Theorie, postulierter Quellkode) und ein Testbed vorgestellt, in dem der Hüllkurvendetektor in ein Messsystem eingebettet wird. Dieses berücksichtigt Parameter in der Messkette und inkludiert neben deren Modell auch eines des gemessenen Signals. Die Simulationen werden in Matlab® implementiert.

Schlüsselwörter: Hilbert-Transformation, Analytische Signal, Gibbsphänomen, Fenster, Diskretisierungsfehler auf Grund von Floating-Point-Variablen, Typ-A-Signal, RFID, Harmonische, nichtlineare Systeme, gaußsches Tiefpassfilter, Kubisches Verstärkermodell, Abtaster, Zeitjitter, Quantisierer, additives weißes gaußsches Rauschen, mittlerer quadratischer Fehler (MSE)

Contents

Preface 1
Motivation and problem statement 2

I Envelope Detector **3**

1 Representation of bandpass signals and the envelope **5**
1.1 Representations of bandpass signals 7
1.1.1 Equivalent baseband signal 8
1.1.2 Canonical representations 9
1.1.3 Dugundji's formulation of the envelope 9
1.2 The discrete-time Hilbert transformer 9

2 Analysis of an envelope detector algorithm **12**
2.1 Concept of analysis 12
2.2 Gibbs phenomenon 13
2.2.1 Generalized cosine window 15
2.2.2 Design of Optimal filters 16
2.2.3 Frequency-domain least-squares filter design 17
2.3 Review: Deterministic signals 18
2.4 Review: Stochastic signals 18
2.4.1 Introduction 19
2.4.2 Two random variables 19
2.4.3 Discrete-time random process 19
2.4.4 Estimation of Moments 20
2.4.5 Distributions 21
2.5 Analysis 21
2.5.1 Type of implementation 22
2.5.2 Analysing of the postulated source code 22
2.5.3 Error of the whole implemented algorithm compared with an implementation in Matlab® 25
2.5.4 Implementation of the Fast Fourier Transformation 27
2.5.5 Specification of the Gibbs phenomenon with a simulation . . 31
2.6 Summary 41

II Influence of the analogue-to-digital conversion chain 43

3 Sampling and bandwidth of an oscilloscope 45
3.1 Sampling techniques . . . 46
3.2 Bandwidth of frontend hardware . . . 46
3.3 Real-world sampling . . . 48

4 Models 50
4.1 Envelope Detectors applied to real-world signals . . . 50
4.1.1 Source generation . . . 51
4.1.2 Measurement chain . . . 53

5 Model implementation 58
5.1 Source model . . . 59
5.2 Measurement path model . . . 61
5.3 Reference path . . . 64
5.3.1 Implementation . . . 64
5.3.2 Error . . . 65
5.3.3 Additional notes to the implementation in Matlab® . . . 65

6 Common setup of source model 67
6.1 Spectrum of the signal produced by the source model. . . . 67
6.2 Example: Agilent's DSO7052A . . . 68

7 Results of simulation and Conclusions 71
7.1 Influence of harmonics to the received signal . . . 71
7.2 Influence of the intermodulation within the oscilloscope . . . 72
7.3 Influence of the sampler . . . 72
7.4 Influence of Gaussian noise . . . 73
7.5 Influence of quantization . . . 73
7.6 Influence of the time jitter of the sampler . . . 75
7.7 Influence of delay of harmonics . . . 75
7.8 Influence of the frequency response of the oscilloscope . . . 75

8 Summary and conclusion 81

A Confidence Interval 82

B Pulse amplitude modulation 83

C Rise time, settle time and overshoot 84

D Source Code **85**
D.1 (I)FFT 85
D.1.1 Header 85
D.1.2 Source file 85
D.2 Hilbert detector 88

List of figures **95**

List of tables **98**

Bibliography **100**

Index **102**

Nomenclature **104**

Preface

I am grateful for the help of my supervisors Dipl.-Ing. Ralph Prestros, Ass.Prof. Dipl.-Ing. Dr.techn. Gerhard Doblinger and Univ.Prof. Dipl.-Ing. Dr.-Ing. Christoph Mecklenbräuker, as well as for discussions with my colleagues Gregor Lasser, Andreas Körner and others.

Special thanks go to my parents Dr. Gerhard and Eva Xaver who support me a lot. Furthermore thanks to my girlfriend Sabrina Fichtinger for proofreading.

In the course of this diploma thesis I learned and re-called many interesting topics, whereas only some topics found the way into it.

> It is true, that no one can essentially cultivate exact science without understanding the mathematics of that science. But we are not to suppose that the calculations and equations that mathematicians find so useful constitute the whole of mathematics. The calculus is but a part of mathematics.
>
> James Clerk Maxwell[1]

Vienna, June 8, 2009

[1] 13^{th} June 1831 – 5^{th} November 1879. He was a Scottish physicist, mathematician and philosopher.

Motivation and problem statement

The topic of this diploma thesis deals with Radio Frequency Indentification (RFID) applications, in particular the test case of card readers for ePassport. ePassport defines a sub-standard for the transmission of data in the near field between passports and readers. Both, ePassport and reader contain conductor loops to use induction for the transmission in the high frequency range. All over the world readers have to be tested to make sure that they comply the standard. The envelope of the signal at the receiver-coil of the reader has to fulfill a shape defined in [ISO08].

In the past usually an oscilloscope[2] has been used to compare the measured signal with the defined signal shape. In ISO/IEC 10373-6:2001 Amendment 7 (Test methods for ePassport) a new solution with source code in the programming language C has been postulated. It uses the HILBERT[3] transform to calculate the envelope of a signal. Nobody had illustrated the reason why another type had been chosen so that the algorithm and the implementation had to be analysed. Some questions came up:

1. Is it better than the previous used method?
2. What are the pros and cons of this detector?
3. How could the implementation be improved?
4. What is the influence of the measurement chain?

Moreover a testbed could identify the influence of parameters like sampling rate, quantization bits or noise to the reliability of the test system. Even though parameters from the ePassport standard are used in this thesis to answer the questions and to simulate the test system, the results can be applied quite generally. The simulations can be adapted easily. Thus this diploma thesis can provide some benefits even for non-RFID applications when skipping some subsections.

Following from above this thesis is splitted into two parts which have different focuses. The first one only deals with the envelope detector itself with pros and cons. The second one investigates an ideal Hilbert envelope detector embedded into a testbed to show some general influences of sampler, lowpass filter characteristic of the measurment system, quantizer, noise, harmonics and amplifier.

[2]The oscilloscope shows the bandpass signal; with adjusting of the time scale one may see the envelope.

[3]David Hilbert, 23th January 1862 – 14th February 1943, mathematician

Part I

Envelope Detector

Part 1 adresses the envelope detector based on Hilbert transform, whereas Part 2 views the envelope detector in a measurement chain. Here we will discuss two different topics:

Chapter 1 – Representation of bandpass signals and the envelope – introduces some theory about representations of bandpass signals with the goal to describe the envelope of a signal.

Chapter 2 – Analysis of an envelope detector algorithm – analyses the implementation of an envelope detector based on the Hilbert transform postulated in [ISO] and imprinted in Appendix D.

1 Representation of bandpass signals and the envelope

Baseband and bandpass. A bandlimited signal with a frequency range from 0 Hz to an upper frequency f_u is called baseband or lowpass signal. This kind of signal is complex-valued in general. In realworld channels it isn't possible to send complex-valued baseband signals, so one has to move it to higher frequencies. These signals are called bandpass signals with a bandwidth between the lowest frequency $f_l >>$ 0 Hz and a highest frequency $f_u > f_l$. Usually bandpass signals are real-valued. In the next sections we will go further into detail.

Unprofessional formulation of an envelope. An oscilloscope is often used to analyze a signal. One can imagine an amplitude modulated signal where the carrier is modified with a cosine over the time. **Figure 1.1** is such an example. Without knowledge one would expect that the first part of an envelope could be the interpolation of the local maxima of the signal. The second part would consist of the interpolation of the local minima. The question come up whether it would be really the best method since every interpolation introduces failures. We will start with one of the first mathematical definitions with time-continuous signals, generalize it and later discuss the problems in time-discrete systems.

Rice's formulation of an envelope of a bandpass signal. A better definition of the envelope of a bandpass signal than an interpolation has been formulated by S. O. Rice which is described in paper [Dug58]. It is demonstrative so that we will present it in this section.

He assumed a periodic signal and expressed it in the form of Fourier series:

$$u(t) = \frac{c_0}{2} + \sum_{n=1}^{\infty} c_n \cos(\omega_n t + \phi_n). \tag{1.1}$$

The constant term can be set to zero if we assume that the signal doesn't have a DC component.

Bandpass signals have a carrier frequency so it is very intuitive to select a frequency q which is called midband frequency. He reformulated the expression to

$$u(t) = \sum_{n=1}^{\infty} c_n \cos((\omega_n - q)t + \phi_n + qt) \tag{1.2}$$

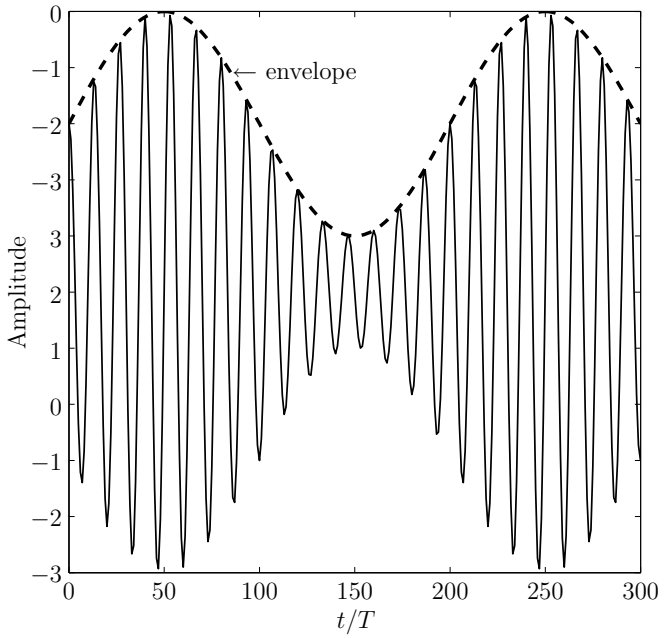


Figure 1.1: Envelope of a modulated cosine wave, called carrier, of $f_c = 15$ Hz with a sinus signal of $f = 1$ Hz.

and used the similarities to circular trigonometric functions:

$$u(t) = \sum_n c_n \cos((\omega_n - q)t + \phi_n) \cos qt - \sum_n c_n \sin((\omega_n - q)t + \phi_n) \sin qt \tag{1.3}$$

$$= u_I(t) \cos qt - u_Q \sin qt \tag{1.4}$$

Now the expression

$$r(t) = \sqrt{u_I^2 + u_Q^2} \tag{1.5}$$

is termed "envelope of u(t) referred to frequency q". This formulation has disadvantages:

1. One must choose a midband frequency, which perhaps will be the carrier frequency, but this isn't clear. The question comes up whether such a special frequency is really necessary.

2. The calculation of $u(t)$ is a huge task. Rice's formula doesn't lead us to an easy implementation.

The following section will present a much better method.

1.1 Representations of bandpass signals

In this section the first effort is to give the derivation of the so called pre-envelope. This will lead us to three mathematical representations of bandpass signals where one of them represents the general definition of a signal envelope.

At the beginning we have to define some mathematical terms and notations which may be slightly different in the literature.

Definition 1 — Fourier transform. The Fourier transform of a signal $x(t) \in \mathbb{C}$ is

$$X(j\omega) = \mathcal{F}\{x(t)\} := \int_{-\infty}^{\infty} x(t)e^{-j\omega t}dt \tag{1.6}$$

and is called frequency spectrum or "the frequency domain representation of the signal $x(t)$". The inverse Fourier transform

$$x(t) = \mathcal{F}^{-1}\{X(j\omega)\} := \frac{1}{2\pi}\int_{-\infty}^{\infty} X(j\omega)e^{j\omega t}d\omega \tag{1.7}$$

is called "the time domain representation of the signal $x(t)$". The notation $x(t) \longleftrightarrow X(j\omega)$ denotes "$x(t)$ correspondences to $X(j\omega)$" and $j := \sqrt{-1}$.

Definition 2 — Convolution. The linear convolution of two functions $x(t)$ and $y(t)$ is defined by

$$x(t) \star y(t) := \int_{-\infty}^{\infty} x(\tau)y(t-\tau)d\tau. \tag{1.8}$$

Definition 3 — Analytic Signal. Let $f(t) \in \mathbb{R}$ be a bandpass signal. An analytic signal is a function $f_+(t) : \mathbb{R} \longmapsto \mathbb{C}$, where $f_+(t) = \mathcal{A}\{f(t)\}$, whose Fourier transform contains only positive frequencies:

$$\mathcal{F}\{f_+(t)\} = 0 \text{ for } \omega < 0 \text{ rad/s}. \tag{1.9}$$

It is sometimes called complex envelope or pre-envelope too. Carefully note that this is not the definition of an analytic function!

Derivation of the analytic signal of a bandpass signal. Suppose a real-valued waveform $u(t) \in \mathbb{R}$ which has a positive and a negative frequency component. The first step is to cut off the negative one. The result has to be multiplied with the factor two to retain the same energy of the signal. With the HEAVISIDE step

function[1]

$$\sigma(j\omega) := \begin{cases} 0 & \omega < 0 \text{ rad/s} \\ \frac{1}{2} & \omega = 0 \text{ rad/s} \\ 1 & \omega > 0 \text{ rad/s} \end{cases} \tag{1.10}$$

we have

$$U_+(j\omega) := 2U(j\omega)\sigma(j\omega). \tag{1.11}$$

Using the inverse Fourier transform this expression is the convolution of

$$2\mathcal{F}\{U(j\omega)\} \star \mathcal{F}^{-1}\{\sigma(j\omega)\}. \tag{1.12}$$

With $\sigma(j\omega) \Longleftrightarrow \delta(t) + j\frac{1}{\pi t}$ we get

$$u_+(t) = u(t) \star \delta(t) + ju(t) \star \frac{1}{\pi t}. \tag{1.13}$$

The latter term is called *Hilbert transform* which is named after David Hilbert. It can be viewed as a linear time invariante (LTI) system where

$$\frac{1}{\pi t} =: h(t) \Longleftrightarrow H(j\omega) = -j\text{sign}(\omega) = \begin{cases} -j & \omega > 0 \text{ rad/s} \\ 0 & \omega = 0 \text{ rad/s} \\ j & \omega < 0 \text{ rad/s} \end{cases} \tag{1.14}$$

is the impulse response and frequency response of the filter ($\text{sign}(x) = x/|x|\text{sign}(0) := 0$). We note that it is a 90° *phase shifter* with $|H(j\omega)| = 1$ for $\omega \neq 0$.

Now we can rewrite our complex-valued signal to

$$u_+(t) = u(t) + j\mathcal{H}\{u(t)\} \tag{1.15}$$

where $\mathcal{H}\{\,\cdot\,\}$ is the Hilbert operator. We see that Equation (1.15) satisfies the definition of the complex-valued analytic signal from above.

1.1.1 Equivalent baseband signal

The analytic signal is still in the bandpass domain. The equivalent baseband signal can be obtained by shifting to lower frequencies. According to the Fourier correspondence

$$e^{-j\omega_0 t}x(t) \Longleftrightarrow X(j(\omega + \omega_0)) \tag{1.16}$$

we get

$$u_e(t) = u_+(t)e^{-j\omega_0 t}. \tag{1.17}$$

$f_0 = \omega_0/2\pi$ is the carrier frequency of the modulated signal.

[1] Since a bandpass signal doesn't have a DC part, $\frac{1}{2}$ at frequency $\omega = 0$ rad/s doesn't hurt.

1.1.2 Canonical representations

Since $u_e(t) \in \mathbb{C}$ it can be expressed as

$$u_e(t) = u_I(t) + j u_Q(t). \tag{1.18}$$

$u_I(t)$ is called the *in-phase* and $u_Q(t)$ the *quadrature* component. The visualization of these components is called the constellation diagram where the abscissa specifies the in-phase term and the ordinate specifies the quadrature term.

1.1.3 Dugundji's formulation of the envelope

In the following the notation $\mathrm{Re}(\,\cdot\,)$ specifies the real part and $\mathrm{Im}(\,\cdot\,)$ the imaginary part of a complex-valued signal. The third respresentation of bandpass signals can be given with the envelope

$$r(t) = |u_+(t)| = \sqrt{\mathrm{Re}^2(u_+(t)) + \mathrm{Im}^2(u_+(t))} \tag{1.19}$$

and the phase

$$\phi(t) = \tan^{-1} \frac{\mathrm{Im}(u_+(t))}{\mathrm{Re}(u_+(t))}. \tag{1.20}$$

$\tan^{-1}(\,\cdot\,)$ indicates the arc tangent. J. Dugundji showed in [Dug58] that this formulation gives the same result like Rice's one, so that $r(t)|_{\mathrm{Dugundji}} \equiv r(t)|_{\mathrm{Rice}}$. The instantaneous frequency is the derivation of the phase:

$$\omega(t) = \frac{\partial}{\partial t}\phi(t). \tag{1.21}$$

With (1.19) and (1.20) the signal can be also rewritten to

$$u(t) = r(t)\cos\left(\omega_0 t + \phi(t)\right) \tag{1.22}$$

which is the common definition in the literature.

For further reading you will find some chapters about Hilbert transform in [Dug58], [pro96], [opp92] and [mit02].

1.2 The discrete-time Hilbert transformer

In the last section we have discussed the definition of the envelope and how to get it from a bandpass signal. The next question is how to implement it in Matlab®[2] or another programming language. One has to sample the signal and to use the discrete-time Hilbert transform to calculate the envelope.

First we have to define the discrete-time Fourier transform.

[2]MATrix LABoratorym, http://www.mathworks.de

Definition 4 — Discrete-time Fourier transform. The discrete-time Fourier transform or frequency spectrum of a discrete-time signal (where $n \in \mathbb{Z}$) $x[n]$ with $\Theta = \omega T_s = 2\pi \dfrac{f}{f_s}$ is

$$X(e^{j\Theta}) = \mathcal{F}\{x[n]\} := \sum_{n=-\infty}^{\infty} x[n] e^{-j\Theta n}. \tag{1.23}$$

f_s is the sampling frequency, T_s is the sample period. The inverse discrete-time Fourier transform is

$$x[n] = \mathcal{F}^{-1}\{X(e^{j\Theta})\} := \frac{1}{2\pi} \int_0^{2\pi} X(e^{j\Theta}) e^{j\Theta n} d\Theta. \tag{1.24}$$

With an abuse of notation we write the same symbol for discrete-time and continuous-time Fourier transform.

With a time delay of n_0 samples the frequency response of the discrete-time Hilbert transformer can be written using (1.14) and

$$X(e^{j\Theta}) = \mathcal{F}\{x(nT)\} = \frac{1}{T} \sum_{k=-\infty}^{\infty} X\left(j\frac{\Theta}{T} + j\frac{2\pi}{T}k\right) \tag{1.25}$$

as[3]

$$H(e^{j\Theta}) = -j\text{sign}(\Theta) e^{j\Theta n_0} = \begin{cases} e^{-j(\Theta n_o + \frac{\pi}{2})} & 0 < \Theta < \pi \\ 0 & \Theta \in \{0, +\pi, -\pi\}\,, n_o \in \mathbb{Z} \\ e^{-j(\Theta n_o - \frac{\pi}{2})} & -\pi < \Theta < 0 \end{cases} \tag{1.26}$$

Using the inverse discrete-time Fourier transform

$$\mathcal{F}^{-1}\{H(e^{j\Theta})\} = \frac{1}{2\pi} \int_{-\pi}^{0} e^{j\Theta(n-n_0)+j\pi/2} d\Theta + \frac{1}{2\pi} \int_0^{\pi} e^{j\Theta(n-n_0)-j\pi/2} d\Theta = \tag{1.27}$$

$$= \frac{j}{2\pi} \int_{-\pi}^{0} e^{j\Theta(n-n_0)} d\Theta - \frac{j}{2\pi} \int_0^{\pi} e^{j\Theta(n-n_0)} d\Theta = \tag{1.28}$$

$$= \frac{1}{2\pi(n-n_0)} \left[\left(1 - e^{-j\pi(n-n_0)}\right) - \left(e^{j\pi(n-n_0)} - 1\right)\right] \tag{1.29}$$

$$= \frac{1}{\pi(n-n_0)} \left(1 - (-1)^{n-n_0}\right) \tag{1.30}$$

[3]Cf. in German [dob08] and in English [ste96]

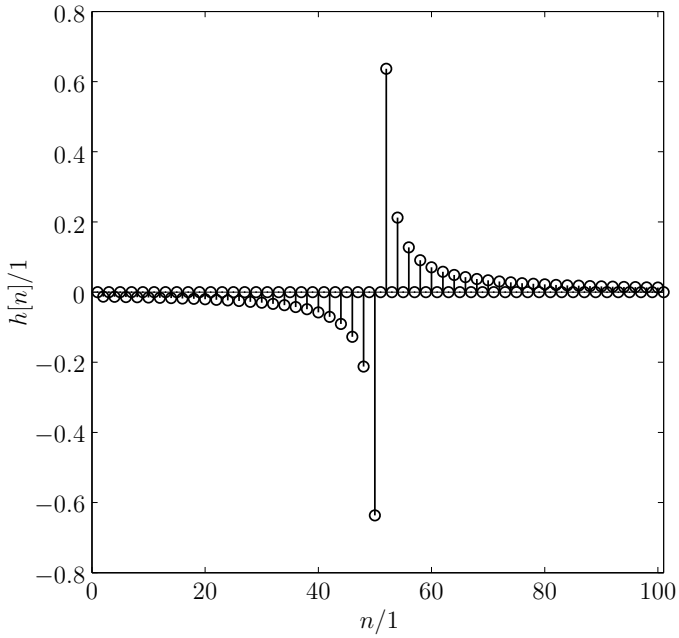


Figure 1.2: Impulse response of a discrete-time Hilbert transformer truncated after $N = 100$ samples with delay of $n_0 = N/2$.

we get the impulse response

$$h[n] = \begin{cases} \dfrac{2}{\pi(n-n_0)} & n - n_0 \text{ odd}, \\ 0 & n - n_0 \text{ even} \end{cases} \tag{1.31}$$

for any $n \in \mathbb{Z}$. The ideal impulse response of (1.31) with $n_0 = 0$ has two important properties,

1. infinite length and
2. acausality,

which brings problems when it has to be implemented as a finite impulse response (FIR) filter. To obtain a finite-length filter one has to truncate the impulse response after N samples which will be discussed in section 2.2 on page 13. The practical FIR filter needs to be causal, so the Function (1.31) has to be delayed by

$$n_0 := \begin{cases} (N-1)/2 & N \text{ odd}, \\ N/2 & N\text{even} \end{cases} \tag{1.32}$$

samples. A practical example is shown in **Figure 1.2**.

2 Analysis of an envelope detector algorithm

In section 1.2 we discussed the discrete-time Hilbert transform, which is an alternative to the commonly used IQ-demodulator for implementations of evelope detectors. In principle there are two possible ways of implementation:

1. *Time domain:* Implementing the impulse response (1.31) with a Finite Impulse Response (FIR) filter[1] or

2. *Frequency domain:* transforming the signal via Discrete Fourier Transform (DFT) into the frequency domain, applying the transfer function (1.26) and using the Inverse Discrete Fourier Transform (IDFT) to go back into the time domain.

The task of this section is to discuss the assets and drawbacks of the algorithm and its implementation in [ISO]. Simulations will be done in Matlab®. The algorithm from the ISO/IEC-paper is written in the programming language C, which will be compiled in our case with the GNU C compiler[2].

This chapter has several sections:

2.1 Concept of Analysis. This section will discuss the parts of our analysis.

2.2 Gibbs phenomenon. Here, the Gibbs phenomenon, which introduces artifacts, will be discussed.

2.3 Deterministic signals and

2.4 Stochastic signals will recall some important definitions.

2.5 Analysis. The last section will include the analysis of the source code.

2.1 Concept of analysis

Following parts have to be analyzed to evaluate the algorithm and its implementation:

[1] Using fast convolution would be possible as well.

[2] The compiler (http://www.gnu.org) is available for most operating systems: DOS, Linux, Windows, MacOS X etc.

Type of implementation. This is the easiest part, a quick analysis of the source code will tell us which type from above is used.

Gibbs phenomenon/Window. This artefact is a result in Method (1) when truncating the impulse response of the Hilbert transformer. In Method (2) the DFT causes it. It will be discussed in Section 2.2.

Fast Fourier Transformation. On the one hand if Method (2) of the above list is used the implementation of the discrete Fourier transform is important since there are many methods of calculating it. On the other hand the convolution of Method (1) would be implemented using a FIR filter or a fast convolution (which uses the discrete Fourier transform to multiply the spectral of the signal with the transfer function of the Hilbert transformer instead of calculating the convolution in the time domain).

Used signals. The main analysis will be done with uniformly distributed random signals (noise) and later with standard signals. In practice Pulse Amplitude Modulation (PAM) with two symbols in the linear signal space[3] is used. The notation is $\{s^{(1)}, s^{(2)}\}$ which reflects the levels of the amplitudes of both symbols. Thus we will focus upon this type of signal. This kind of modulation is also called Amplitude Shift Keying (ASK) or On-Off-Keying (OOK) if $s^{(2)} \equiv 0$. The calculation of the envelope will be compared with a program in Matlab® and the reference envelope to discuss the difference. This will be done visually using the autocorrelation, energy spectrum density or power spectrum density or with numerical values like mean, variance or norm.

Source code analysis. The analysis of the implementation in [ISO] should also involve errors and problems in the code of the C program.

2.2 Gibbs phenomenon

This is an important characteristic so we go further into details.

Discontinuities. The Gibbs phenomenon, which is named after Josiah Willard Gibbs, exists at jump discontinuities of time-continuous signals since it is not possible to approximate discontinuous functions with continuous functions. The results are *oscillations* (also called *ripples*) and *overshoots* are close to an edge. *The phenomenon exists in time-discrete systems, too, except for (jump) discontinuities of periodic time-discrete signals that don't cause it (Cf. with[dob08] at page 21ff).*

[3]The signal space is a Hilbert space and the graphical visualisation is called constellation diagram.

Windows. If implementation (1) is used, the impulse response of the Hilbert transformer has to be truncated because it has infinite length. The DFT implementation (2) has the same phenomenon since it is applied over a finite number of samples and is called *Leakage*. Let $h[n]$ be the impulse response and $w[n]$ a so called window sequence:

$$h_t[n] = h[n]w[n]. \tag{2.1}$$

In the theoretical case we have an infinite window, thus[4] $w[n] = 1 \ \forall n$. In the frequency domain this corresponds to

$$H_t(e^{-j\Theta}) = \delta_{2\pi}(\Theta) \star H(e^{-j\Theta}). \tag{2.2}$$

With the identity $X(e^{-j\Theta}) \equiv X(e^{-j\Theta}) \star \delta_{2\pi}$ we note that there is no modification of the signal. In practice one has a window with finite length, which is in the simplest case a rectangle

$$w[n] = w_R[n] := \begin{cases} 1 & 0 \leqslant n \leqslant M \\ 0 & \text{otherwise,} \end{cases} \tag{2.3}$$

where $M + 1$ is the length of the window. Thus $h_t[n]$ becomes

$$h_t[n] = \begin{cases} h[n] & 0 \leqslant n \leqslant M \\ 0 & \text{otherwise.} \end{cases}. \tag{2.4}$$

Two jump discontinuities follow which result in the presence of large oscillations. In the frequency domain the rectangle window is

$$W(e^{-j\Theta}) = \mathcal{F}^{-1}\{w[n]\} = \frac{\sin\left((M+1)\dfrac{\Theta}{2}\right)}{\sin\dfrac{\Theta}{2}} e^{-j\Theta\dfrac{M}{2}} \tag{2.5}$$

where $W(e^{-j0}) = 1$. Thus the frequency response of the truncated signal is

$$H_t(e^{-j\Theta}) = \frac{\sin\left((M+1)\dfrac{\Theta}{2}\right)}{\sin\dfrac{\Theta}{2}} e^{-j\Theta\dfrac{M}{2}} \star H(e^{-j\Theta}). \tag{2.6}$$

We note that the convolution in (2.6) modifies the original signal. The window part is responsible for that so we have to look closer at the frequency response of the window.

In **Figure** (2.1) we see a main lobe close to $\Theta = 0$. If M increases, the width of the lobes decreases and the lobes tend to $\delta_{2\pi}(\Theta)$. Thus M should be as large as

[4] $\forall$... "for all"

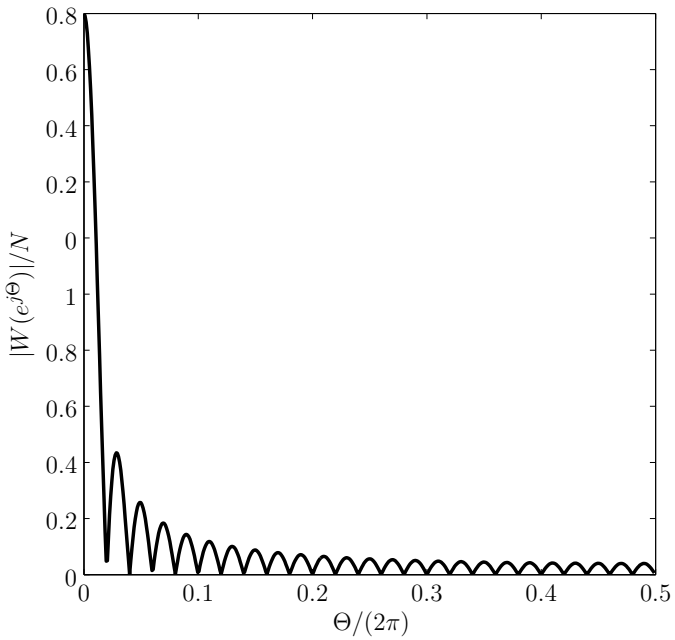


Figure 2.1: Frequency response of a rectangle window with the length of 50.

possible. On the other hand the computation (time) increases, so we wish a small M.

Note that since the area under the lobes is constant the amplitude of the ripples will remain constant too! Thus if M increases, the oscillations are faster but aren't reduced. The amplitudes of the ripples don't become smaller (approximately 9% of the jump).

Frequency response of a time-discrete Hilbert transform. At page 11 we have discussed about the Hilbert transformer and have seen the impulse response. Owing to the lobes of the transfer function of the window sequence we are expecting ripples in the frequency domain. **Figure 2.2** shows the corresponding transfer function.

2.2.1 Generalized cosine window

To avoid this problem other windows can be used which should have a *smoother slope* and the *maxima of the side lobes should be as small as possible.* A very important class of such windows is the *generalized cosine window*:

$$w[n] := w_c[n] = \begin{cases} a - b\cos\frac{2\pi n}{N-1} + c\cos\frac{4\pi n}{N-1} & 0 \leqslant n \leqslant N-1 \\ 0 & \text{elsewhere} \end{cases} \quad a, b, c \in \mathbb{R}. \qquad (2.7)$$

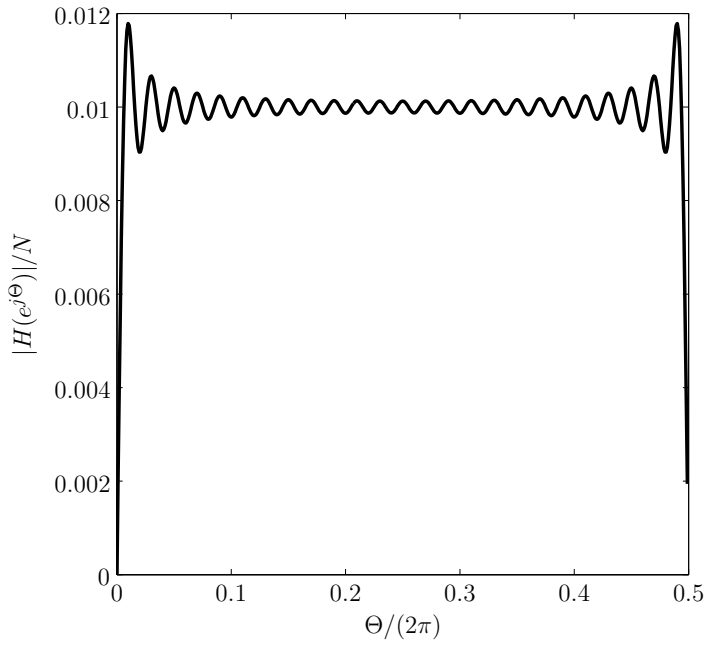


Figure 2.2: Transfer function of a discrete-time Hilbert transformer truncated after $N = 100$ samples with delay of $n_0 = N/2$ (rectangular window).

window	a	b	c
rectangle	1	0	0
Hann	0.5	0.5	0
Hamming	0.54	0.46	0
Blackman	0.42	0.5	0.08

Table 2.1: Parameters a, b and c of equation (2.7) of some important windows.

Equation (2.7) generalizes many important and well-known windows shown in Table 2.1. Other more complex windows are the *Kaiser window* and the *Chepyshev window* where one parameter is the amplitude of the ripples.

2.2.2 Design of Optimal filters

With the *Parks-McClellan algorithm* it is possible to design equi-ripple (ripple with constant peak amplitude) linear-phase FIR filters which minimize the peak absolute value of the weighted error, called *Chebyshev criterion*,

$$\epsilon := \arg\max_{\omega} |E(\omega)| = \arg\max_{\omega} |W(e^{j\Theta}) \left(\tilde{H}(e^{j\Theta}) - H(e^{j\Theta}) \right)| \tag{2.8}$$

where $W(e^{j\Theta})$ is the weighting function, $\tilde{H}(e^{j\Theta})$ the desired frequency response of the filter and $H(e^{j\Theta})$ the approximated filter. In Matlab® the function `firpm()`

can be used. The length of those filters is minimal in this sense.

2.2.3 Frequency-domain least-squares filter design

Another filter design method is the so called *Frequency-Domain Least-Squares* (FDLS) method. In this paragraph the idea behind FDLS will be briefly presented.

The frequency response of a filter consists of magnitude $A[k]$ and phase $\phi[k]$ which will be denoted by A_k and phase ϕ_k. k is the frequency index. Therefore at a frequency ω_k the amplitude of the output is multiplied by A_k and the phase is shifted by ϕ_k. If the input signal $x_k[n] = \cos(\omega_k T_s n)$ with the sample period T_s the output signal will be $y_k[n] = A_k \cos(\omega_k T_s n + \phi_k)$.

Let us assume that the filter is causal and that there exists a (finite) difference equation

$$y_k[n] = -a_1 y_k[n-1] - \cdots - a_D y_k[n-D] + b_0 x_k[n] + \cdots + b_N x_k[n-N] =$$

$$\begin{pmatrix} -y_k[n-1] & \cdots & -y_k[n-D] & x_k[n] & \cdots & x_k[n-N] \end{pmatrix} \begin{pmatrix} a_1 \\ \vdots \\ a_D \\ b_0 \\ \vdots \\ b_N \end{pmatrix}, \quad (2.9)$$

where a and b factors are real-valued filter coefficients and y and x are output and input signals. D and N are real-valued and has to finite since the equation has to be finite. Thus the output value $y_k[n]$ consists of the current and past input and output signals.

It can be formulated in matrix notation by

$$\begin{pmatrix} y_1[0] \\ \vdots \\ y_M[0] \end{pmatrix} = \begin{pmatrix} -y_1[-1] & \cdots & -y_1[-D] & x_1[0] & \cdots & x_1[-N] \\ \vdots & & \vdots & \vdots & & \vdots \\ -y_M[-1] & \cdots & -y_M[-D] & x_M[0] & \cdots & x_M[-N] \end{pmatrix} \begin{pmatrix} a_1 \\ \vdots \\ a_D \\ b_0 \\ \vdots \\ b_N \end{pmatrix} \quad (2.10)$$

with $M \in \mathbb{N}$. Equation (2.10) can be written as

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{a}.$$

Since this is an overdetermined system $\boldsymbol{a}$ can only be calculated by the pseudo-inverse; hence the filter coefficients

$$\boldsymbol{a} \approx (\boldsymbol{X}^T \boldsymbol{X})^{-1} \boldsymbol{X} \boldsymbol{y}, \quad (2.11)$$

where $\cdot^T$ denotes the transposed matrix. In Matlab® the function `firls()` can be used. For deeper understanding one good paper is [Ber07].

2.3 Review: Deterministic signals

In this section we recall some important definitions and notations which we will use in further sections. You may want to skip it.

Definition 5 — Discrete Fourier Transform and Fast Fourier Transform. The Discrete Fourier Transform (DFT) of a time-discrete sequence with N complex-valued numbers is another complex-valued sequence of length N and is defined as

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j2\pi kn/N} \quad 0 \leqq k \leqq N-1. \tag{2.12}$$

Similarly, the Inverse Discrete Fourier Transform (DFT) becomes

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] e^{j2\pi kn/N} \quad 0 \leqq k \leqq N-1. \tag{2.13}$$

There are some very time-saving algorithms which implement the DFT as Fast Fourier Transformation (FFT).

Definition 6 — Temporal correlation function and autocorrelation. The temporal correlation of two complex-valued signals $x_1[n]$ and $x_2[n]$ is

$$R_{x_1,x_2}[n] := \sum_{k=-\infty}^{\infty} x_1^*[k] x_2[n+k]. \tag{2.14}$$

If $x_1[k] \equiv x_2[k]$ then R_{xx} will be called the temporal autocorrelation of a signal.

Definition 7 — Energy spectrum density. The Fourier transform of the temporal autocorrelation of a signal x,

$$S_{xx}(e^{j\Omega}) := \mathcal{F}\{R_x\} = |X(e^{j\Omega})|^2, \tag{2.15}$$

is called energy spectrum density.

2.4 Review: Stochastic signals

Since we will use stochastic signals in this paper to show several probabilities of the algorithm we have to recall and define some elementary functions and notations. With an abuse of notation random variables will be written with upper-case symbols (i.e. X) in the following. You may want to skip it.

2.4.1 Introduction

Definition 8 — Cumulative distribution and probability density function. The cumulative distribution function (cdf) of a random variable X is defined as the probability that $X \leqq x$:

$$F_X(x) := P\{X \leqq x\}. \tag{2.16}$$

The probability density function (pdf) is written as

$$f_X(x) := \frac{d}{dx} F_X(x). \tag{2.17}$$

The pdf can be used to calculate the probability of the event that the variable X is in the interval $(a; b]$:

$$P\{a < X \leqq b\} = \int_a^b f_X(x) dx. \tag{2.18}$$

Definition 9 — Mixed moments. The moments of a complex-valued random variable X are

$$m_X^{(k)} := E\{X^k\} = \int_{-\infty}^{\infty} x^k f_X(x) dx. \tag{2.19}$$

2.4.2 Two random variables

Definition 10 — Mixed moments. The mixed moment of two complex-valued random variables X_1 and X_2 for $k_1, k_2 \in \mathbb{Z}$ is

$$m_{X_1,X_2}^{(k_1,k_2)}; = E\{X_1^{k_1} X_2^{k_2 *}\} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} x_1^{k_1} x_2^{k_2 *} f_{X_1,X_2}(x_1, x_2) dx_1 dx_2. \tag{2.20}$$

Definition 11 — Correlation function. The correlation of two random variables X_1 and X_2 is

$$R_{X_1,X_2} := E\{X_1 X_2^*\} = m_{X_1,X_2}^{(1,1)}. \tag{2.21}$$

2.4.3 Discrete-time random process

A discrete-time random process is a sequence of random variables $X[n]$ with $n \in \mathbb{Z}$:

$$...X[-1], X[0], X[1], X[2], X[3]... \tag{2.22}$$

Definition 12 — Autocorrelation function. The autocorrelation of two random variables $X[n_1]$ and $X[n_2]$ is

$$R[n_1, n_2] := R_{X[n_1],X[n_2]} = E\{X[n_1]^* X[n_2]\} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} x_1^* x_2 f_{X[n_1],X[n_2]}(x_1, x_2)dx_1 dx_2. \tag{2.23}$$

Wide-sense stationarity. In the following we assume that the signals will be wide-sense stationary[5], so that the second-order statistic property $R_X[n_1, n_2]$ only depends on the difference $m = n_2 - n_1$:

$$r_X[m] := R_X[n_1, m + n_1]. \tag{2.24}$$

Definition 13 — Power spectrum density. The power spectrum density (pds)

$$S_X(e^{j\Theta}) := \mathcal{F}\{r_X[m]\}_{m\to\Theta} = \sum_{m=-\infty}^{\infty} r_X[m]e^{-j\Theta m}. \tag{2.25}$$

The latter two formulas are correct for estimations too.

2.4.4 Estimation of Moments

The pdf of a random variable is often unknown. But if the random experiment is performed N times and N samples are observed then all moments can be estimated. Estimations are denoted with a hat $\hat{\cdot}$.

Definition 14 — Estimated sample moments. Using the observed variables x_n the estimated sample moment $m_X^{(k)}$

$$\hat{m}_X^{(k)} := \frac{1}{N}\sum_{n=1}^{N} x_n^k. \tag{2.26}$$

The estimated central sample moments $m_{X-\mu_X}^{(k)}$ are defined as

$$\hat{m}_{X-\mu_X}^{(k)} := \frac{1}{N}\sum_{n=1}^{N}(x_n - \hat{\mu}_X)^k. \tag{2.27}$$

[5]Strict-sense stationarity means that all order statistic properties depend only on the difference $n_2 - n_1$.

Two very important moments are the sample mean and the sample variance:

$$\hat{\sigma}_X^2 := m_{X-\mu_X}^{(2)}, \tag{2.28}$$

$$\hat{\mu}_X := m_X^{(1)}. \tag{2.29}$$

Definition 15 — Sample autocorrelation. In the case of wide-sense stationarity the estimated temporal sample autocorrelation $\hat{r}_X[m]$ is defined as

$$\hat{r}_X[m] := \frac{1}{N}\sum_{n=1}^{N} x_n x_{n+m}^*. \tag{2.30}$$

Definition 16 — Sample power spectrum density. The sampled power spectrum density (pds)

$$\hat{S}_X(e^{j\Theta}) := \mathcal{F}\{\hat{r}_X[m]\}_{m\to\Theta} = \sum_{m=-\infty}^{\infty} r_X[m]e^{-j\Theta m} \tag{2.31}$$

and is also called periodogram.

2.4.5 Distributions

There are many time-continuous pdfs so in this subsection two very important pdfs will be recalled which will be usefull in the following sections.

Definition 17 — Uniform distribution. A random signal X is uniformly distributed over the interval $[a; b]$ when

$$f_X[n] = \begin{cases} \frac{1}{b-a} & n \in [a; b] \\ 0 & \text{otherwise.} \end{cases} \tag{2.32}$$

Definition 18 — Normal/Gaussian distribution. With the mean μ_X and the variance σ_X^2 the pdf of a Gaussian random variable is

$$f_X(x) = \frac{1}{\sqrt{2\pi\sigma_X^2}} e^{\frac{(x-\mu_x)^2}{2\sigma_x^2}} \tag{2.33}$$

and is denoted as $X \sim \mathcal{N}(\mu_X, \sigma_X^2)$.

2.5 Analysis

In the folowing the implementation of the envelope detector will be analyzed and stochastic properties of the source code will be estimated.

2.5.1 Type of implementation

The source code of [ISO] uses the algorithm of Method (2) whose block diagram is shown in **Figure 2.3**. Note that sampled time-discrete signals are used, where $u[n] = u(nT)$ and $r[n]$ correspond to Equation (1.19) at Page 9. $n \in \mathbb{Z}$ is the time index and T the sample period. The sampling of the real-world signal will be discussed in Part II.

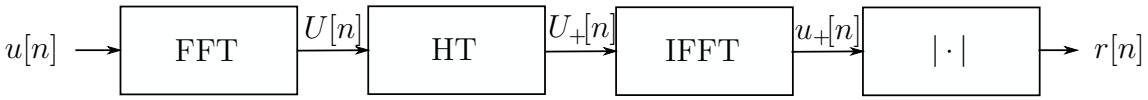


Figure 2.3: Block diagram of the algorithm under evaluation as described in [ISO]. Upper-case symbols represent signals in the frequency domain.

2.5.2 Analysing of the postulated source code

The flowchart of the source code is drawn in **Figure 2.4**. Bold symbols represent vectors or signal sequence. Note the correspondence of

$$u(nT) = u[n] \longleftrightarrow \boldsymbol{u} = \begin{pmatrix} \vdots \\ u[-1] \\ u[0] \\ u[1] \\ \vdots \end{pmatrix}. \tag{2.34}$$

The source code expects that an input file with 5000 samples is available. The input file contains one tuple of time and amplitude per line. If there are less than 5000 samples available there is no error message. The information of time isn't used and thus could be omited.

The code uses a radix 2 Fast Fourier Transform (FFT which is a fast implementation of the DFT) so it can only use data of length 2^n, where $n \in \mathbb{N} \setminus \{1, 2\}$. Thus it uses the FFT of length 8192.

Instead of 8192 values the program writes 5000 tuples in an output file. So the interval $(5000, 8192]$ is cut away. If a step exists at the beginning of the signal, the last samples of the envelope include artifacts from the Gibbs phenomenon which are thrown away. This may be neglected since these are some artifacts from the Gibbs phenomenon.

Since a vector of length 5000 is used to calculate the envelope, the DFT leads to a rectangular window. We have seen in a previous section that this is the worst type of a window and results in ripples with high amplitudes.

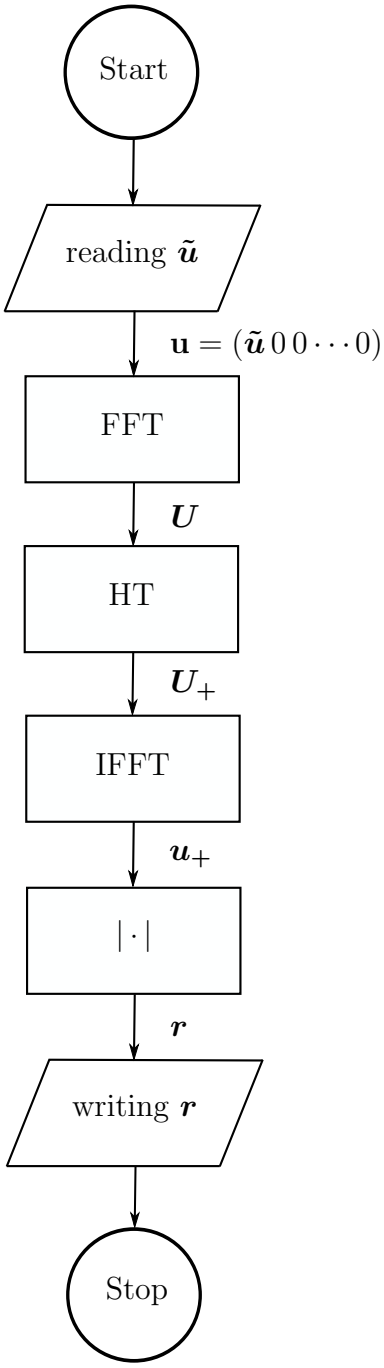


Figure 2.4: Flowchart of the implemented C source code. A vector $\tilde{\mathbf{u}} = (u[0]u[1]u[2]\cdots u[N])$ with $N \leqq 5000$ samples is read from a file or other stream. First zeros are added (*Zero-padding*) to get a vector $\mathbf{u} = (\tilde{\boldsymbol{u}}\,0\,0\cdots 0)$ of length $8192 \equiv 2^{13}$. Applying the Fast Fourier Transform (FFT), the Hilbert transform (HT), the Inverse Fast Fourier Transform (IFFT) and the modulo operator gives the evelope sequence $\boldsymbol{r}$ which is written to a output file or stream.

The program doesn't include code to support more than 5000 samples or a stream of samples. This feature could be done using Overlap-add or Overlap-save, cf. with [dob08].

Note that if signal of length 5000 is used the calculation will be able to be made *after* all samples are available. Thus a FIR filter could be better.

To mention the code style there are many global variables. Thus using it in projects is Gordian. The style looks like a one-to-one conversion from an old Fortran program to a C.

2.5.3 Error of the whole implemented algorithm compared with an implementation in Matlab®

Analysis of errors compared with an implementation in Matlab®. First we are interested in errors of the algorithm in [ISO] compared with an alternative implementation in Matlab®. This software is used as a reference since it is widely-used and it accesses the so called Fastest Fourier Transform in the West[6] (FFTW). It is an open source portable software library under GPL and it's produced norm of the error[7] is smaller than 10^{-17}.

Our analysis will contain following steps, cf. with **Figure 2.5**:

1. The error of the code will be calculated using noise as input signal of $L = 5000$ different realizations. As output we want to get the temporal histogram from the mean of L realizations.

2. As input sequence for a realization l we are using a uniformly distributed random vector

$$\mathbf{X}_l = (X_l[0]X_l[1]X_l[2]...X_l[N])^T \quad 1 \leqq l \leqq L \tag{2.35}$$

   with the length of $N = 5000$ samples. The Matlab® function `rand` is used to get these uniformly distributed vectors $\mathbf{X}_l$ with values in the interval $[-0.5; 0.5]$.

3. This signal will be used as input sequence for the implementation in [ISO] and the "ideal" one in Matlab®.

4. We will call the difference of the output[8] $\mathbf{\Delta Y}_l = \mathbf{Y}_{l,\mathrm{M}} - \mathbf{Y}_{l,\mathrm{C}}$ of both algorithms the *error of [ISO]*.

5. The mean of the realizations will be calculated, thus we get a vector $\hat{\boldsymbol{\mu}}_{\boldsymbol{\Delta Y}}$.

6. The histogram of $\hat{\boldsymbol{\mu}}_{\boldsymbol{\Delta Y}}$ is used to estimate and specify the probability density function (pdf) of the signal.

[6] The design and implementation is discussed in [FJ05]. The homepage can be found at `http://www.fftw.org/`.

[7] We will specify it below in Subsection 2.5.4.

[8] The subscript C specifies the output signal of the Code presented in [ISO], the subscript M the one from the Matlab® version.

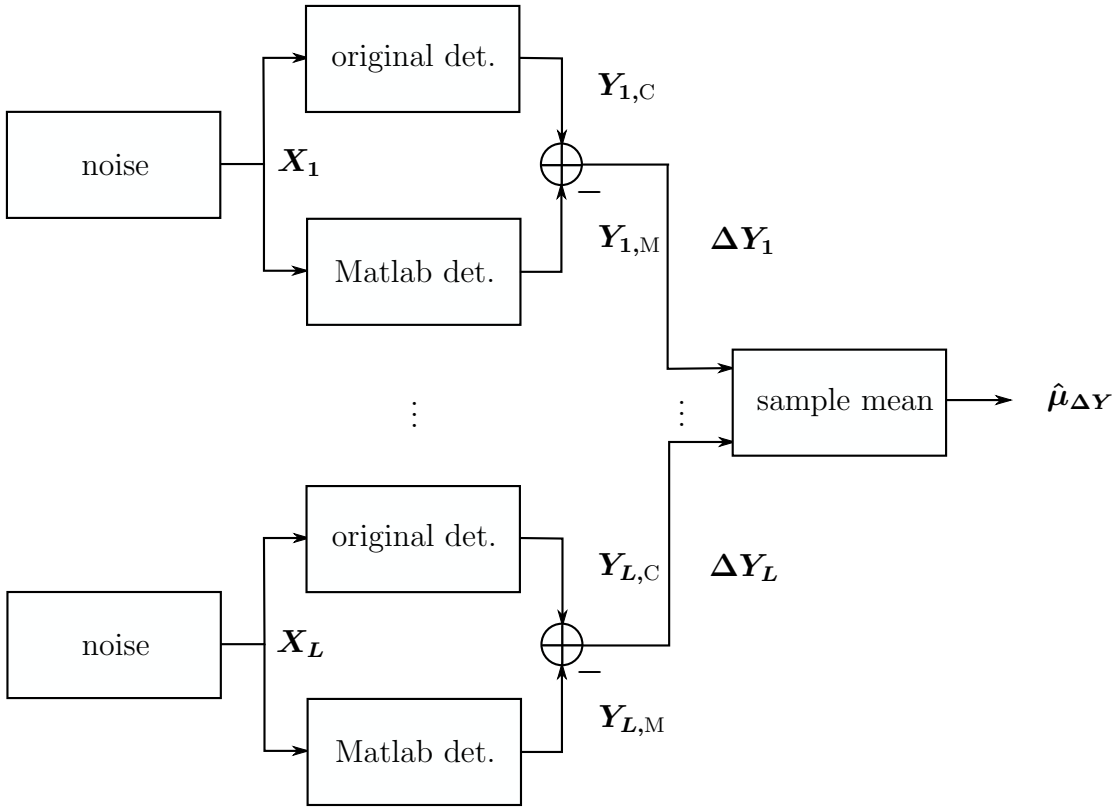


Figure 2.5: Block diagram of the error measurement of the code in [ISO]. It shows L branches from L realisations where the errors of the C-code against a Matlab® version are calculated. From the results we get the sample mean $\hat{\boldsymbol{\mu}}_{\boldsymbol{\Delta Y}}$ which is still a vector. The histogram of this sample mean is plotted in Figure 2.6.

Results. At first step the *estimated sample mean* of realizations is calculated:

$$\hat{\boldsymbol{\mu}}_{\boldsymbol{\Delta Y}} = \frac{1}{L}\sum_{l=1}^{L} \boldsymbol{\Delta Y}_l. \tag{2.36}$$

The number of calculated realizations $L = 1000$. **Figure 2.6** presents the histogram and estimated pdf of the error vector. The estimated sample mean is calculated with

$$\hat{\mu} = \frac{1}{N}\sum_{n=1}^{N} (\hat{\boldsymbol{\mu}}_{\boldsymbol{\Delta Y}})_n \tag{2.37}$$

and the estimated sample variance with

$$\hat{\sigma}^2 = \frac{1}{N}\sum_{n=1}^{N} ((\hat{\boldsymbol{\mu}}_{\boldsymbol{\Delta Y}})_n - \hat{\mu})^2. \tag{2.38}$$

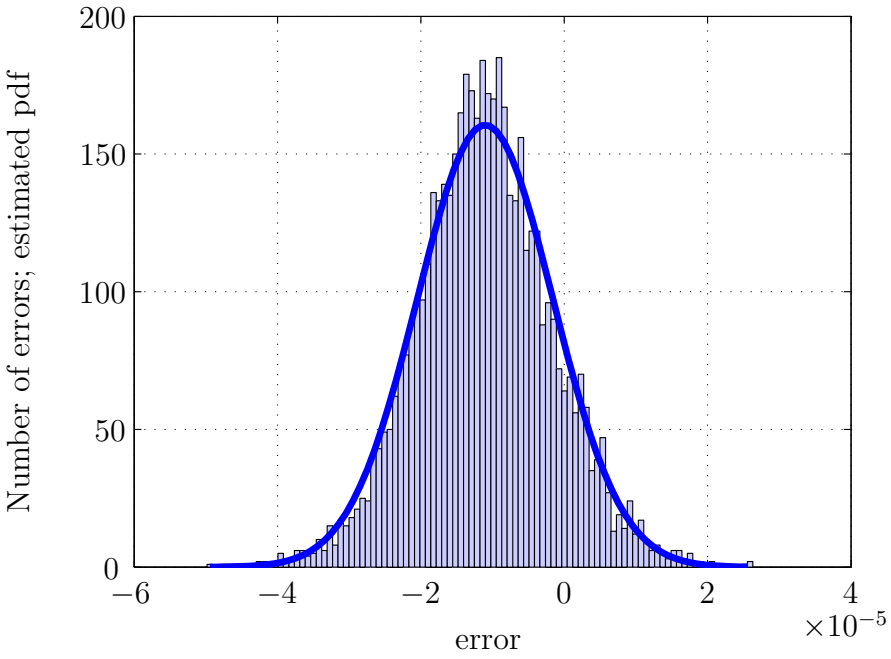


Figure 2.6: Histogram of the resulting noise after $L = 5000$ iterations and estimated probability density function of the continuous signal.

The notation $(\cdot)_n$ specifies the n^{th} element of the vector. With the result

$$\hat{\mu} = -1.1041 * 10^{-5} \text{ and} \tag{2.39}$$

$$\hat{\sigma} = 9.4577 * 10^{-6} \tag{2.40}$$

follows the estimated confidence intervals (cf. with Appendix A) in Table 2.2.

Confidence interval	Systematic error $\hat{\mu}$	Pseudo-random error
95%	$-1.1041 * 10^{-5}$	$\pm 1.2121 * 10^{-5}$
99.99999980%	$-1.1041 * 10^{-5}$	$\pm 5.6746 * 10^{-5}$

Table 2.2: CI of the defined error of the code under evaluation.

The results demonstrate a rather big error. Since the heart of the implementation of the envelope detector is the Fast Fourier Transform we will analyze this part in the next subsection.

2.5.4 Implementation of the Fast Fourier Transformation

Error of the FFT. One method to test an implementation of IFFT and FFT is to calculate

$$y[n] = \mathcal{F}^{-1}\{\mathcal{F}\{x[n]\}\}. \tag{2.41}$$

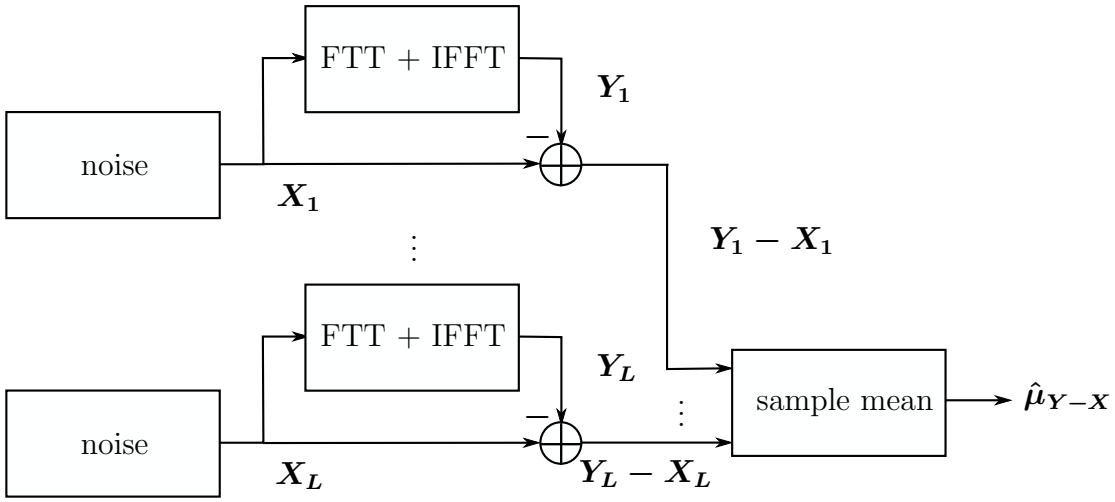


Figure 2.7: Block diagram of the error measurement of a FFT and IFFT routine used in this paper.

Normally $y[n] \equiv x[n]$ if no error exists. In the following two properties are shown which illustrate the error $y[n] - x[n]$.

As input vector we are using a vector of length 1024 which is uniformly distributed noise like in the last subsection. The error $y[n] - x[n]$ is calculated 1000 times to obtain later the estimated mean vector $\hat{\boldsymbol{\mu}}_{\boldsymbol{Y-X}}$ of the $L = 1000$ realizations:

$$\hat{\boldsymbol{\mu}}_{\boldsymbol{Y-X}} = \frac{1}{L}\sum_{l=1}^{L}(\mathbf{Y_l} - \mathbf{X_l}) \tag{2.42}$$

Figure 2.7 shows the corresponding blockdiagram.

From $\hat{\boldsymbol{\mu}}_{\boldsymbol{Y-X}}$ we get the estimated power spectrum density. On the top **Figure 2.8** shows the estimated psd[9] of [ISO] and at the bottom the one from Matlab® (FFTW). The difference of both curves is approx. 200 dB!

We can also specify the norm of the error:

$$||\hat{\boldsymbol{\mu}}_{\boldsymbol{Y-X}}||_2 := \sqrt{\sum_{n=1}^{N}|(\hat{\boldsymbol{\mu}_{Y-X}})_n|^2} = 9.7691 \cdot 10^{-07} \tag{2.44}$$

Cause of the error. Analysing the source code of the file `fftrm.c` according to [ISO] leads us to the error. All variables of the FFT routine use the data type `float`

[9] A variable x in dB (Decibel) is defined as

$$\frac{x}{\text{dB}} := 10\log\left(\frac{x}{1}\right). \tag{2.43}$$

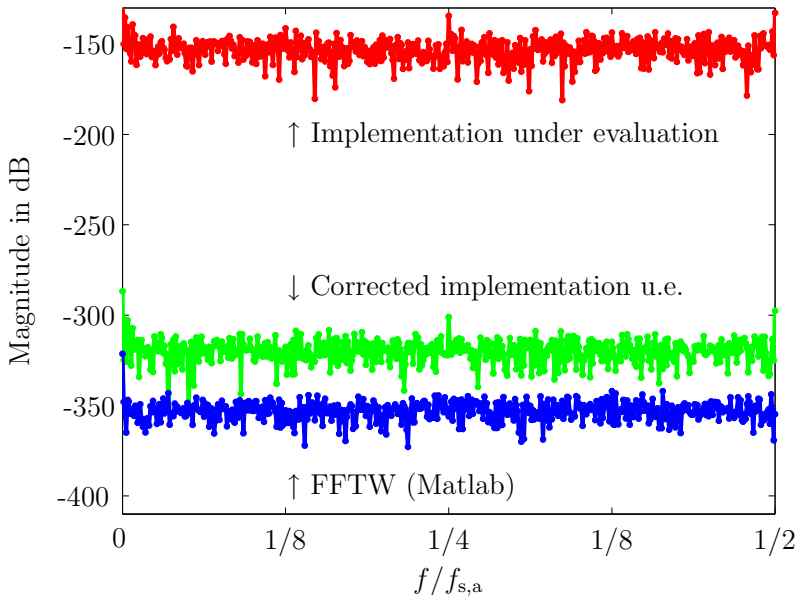


Figure 2.8: The curves show the estimated power spectrum densities from the errors of the three FFT implementations. The difference between the top and the middle curve averages 166 dB, i.e. correcting the float-double bug decreases the bug about 166 dB!

(single-precision). The parameters of the functions however are of type `double` (double-precision). Casting a double-precision type to single-precision type causes heavy errors. The next paragraph explains this behavior.

Single-precision and double-precision floating-point sets. Defined in the paper [flo85] of IEEE, a single- or double-precision floating-point number x is represented by

$$x = (-1)^s \cdot 2^{e-b} \cdot m \tag{2.45}$$

where $s \in \{0, 1\}$ is the sign, e the exponent, m the mantissa and b the exponent bias. The correspond exponent bias and the length of all bit-fields are written in **Table 2.3**. The mapping $\tau : \mathbb{Q}_{64} \to \mathbb{Q}_{32}$ is surjective and therefore produces errors. On the one hand the mantissa has to be rounded, on the other hand a run-time overflow can occur since the length of the exponent changes.

If all `float` declarations are changed to `double` and the simulation from above is repeated one gets the middle curve of **Figure 2.8**. It shows that the improved FFT routine is much better now. The norm of the error

$$||\hat{\boldsymbol{\mu}}_{\boldsymbol{Y}-\boldsymbol{X}}||_2 = 4.7871 \cdot 10^{-15}. \tag{2.46}$$

Discretization error of multiplications and additions of floating-point variables. As we have observed even FFTW introduces errors. The reason are round-off errors

Properties	`float`	`double`
total length (in bit)	32	64
length of sign s (in bit)	1	1
length of exponent e (in bit)	8	11
length of fraction m (in bit)	23	52
exponent bias b	127	1023
set $\{(-1)^s \cdot 2^{e-b} \cdot m; \text{NaN}; \text{INF}\}$	$\mathbb{Q}_{32}$	$\mathbb{Q}_{64}$
volume of set	4 294 967 296	18 446 744 073 709 551 616

Table 2.3: Difference of `float` and `double` data type. NaN denotes "not a number", INF "infinitiy".

and chopping in the computation of the Discrete Fourier Transform via the Fast Fourier Transform algorithm owing to the use of floating-point variables. The set of double-precision floating-point 64-bit values $\mathbb{Q}_{64} \subset \mathbb{Q} \subset \mathbb{R}$. If two variables $x_1 \in \mathbb{Q}_{64}$ and $x_2 \subset \mathbb{Q}_{64}$ are multiplied,

$$x_1 x_2 = \left[(-1)^{s_1} 2^{e_1-1023} m_1\right] * \left[(-1)^{s_2} 2^{e_2-1023} m_2\right] = (-1)^{s_1+s_2} 2^{e_2+e_1-2026} m_1 m_2, \quad (2.47)$$

in general $m_1 m_2$ has double number of digits. Thus the product $x_1 x_2 \notin \mathbb{Q}_{64}$ in general and has to be rounded or chopped. The relative error ε_* is defined in [KL71] by

$$\text{fl}\{x_1 x_2\} = (x_1 x_2)(1 + \varepsilon_*) \quad (2.48)$$

where $\text{fl}\{\cdot\}$ denotes the result using floating-point operations.

The sum of x_1 and x_2

$$x_1 + x_2 = \left[(-1)^{s_1} 2^{e_1-1023} m_1\right] + \left[(-1)^{s_2} 2^{e_2-1023} m_2\right] \quad (2.49)$$

isn't in the subset $\mathbb{Q}_{64}$ too since $2^{e_1-1023} \neq 2^{e_2-1023}$ in general. Thus the mantissa m_1 has to be right shifted by $(e_2 - e_1)$ places and introduces an error ε_1 since the last digits are rounded or chopped, as the case may be:

$$m_1 2^{-(e_2-e_1)} + \varepsilon_1 \quad (2.50)$$

The left term can introduce another error ε_2 when a runtime overflow occurs (right shift $L = -1$) or a renormalization is needed (left shift by $L \geqq 0$):

$$2^{-L} \left[2^L \left[(-1)^{s_1} m_1 2^{-(e_2-e_1)} + (-1)^{s_1} \varepsilon_1 + m_2\right] + \varepsilon_2\right] \quad (2.51)$$

We define the relative error ε_+ by

$$\text{fl}\{x_1 + x_2\} = (x_1 + x_2)(1 + \varepsilon_+) \equiv (-1)^{s_2} 2^{-L} \left[2^L \left[(-1)^{s_1} m_1 2^{-(e_2-e_1)} + \varepsilon_1 + m_2\right] + \varepsilon_2\right] 2^{e_2-1023}. \quad (2.52)$$

Since every implementation of FFT contains many additions and multiplications errors occur. In the following these errors of floating-point operations will be neglected. More results of errors in floating-point arithmetic can be found in [KL71].

Another general view provides [pro96]. The paper [Ram71] is of interest, which derives upper bounds for the error and [TL77] describes a statistical model for round-off errors in floating point FFTs.

2.5.5 Specification of the Gibbs phenomenon with a simulation

The second cause of an error of the used Hilbert transform is the Leakage. The error occurs with the real-world Hilbert transform filter since the impulse response of the infinite Hilbert transform filter has to be cut off to make it finite. This is similar to multiplying the impulse response with a rectangle window. If the detector is realized in the frequency domain like in Annex 7 of [ISO] the FFT causes the same error. We have seen that this Gibbs phenomenon can be reduced with other windows or an Optimal FIR filter (only in the time domain).

Note that the following specified error isn't because of a bad code in Annex 7 of [ISO]. Instead we will show that a general modification of the used Hilbert transform could improve the detection of the envelope.

Now two ways will be simulated and compared with the case of a rectangular window:

1. Optimal FIR filter
2. Blackman window

For the simulations a real-world example of an RFID transmission will be used with an ideal rectangular baseband pulse and an idealized sampler.

Optimal FIR Filter. The Optimal FIR filter which will be utilized in the simulation below uses the Parks-Mc-Clellan algorithm. The desired periodic[10] frequency response

$$|\tilde{H}(e^{j\Omega})| := \begin{cases} 1 & 0.015\pi \leqq |\Omega| \leqq 0.985\pi \\ 0 & \Omega = 0 \text{ and } \Omega = \pm\pi \\ \text{undefined} & \text{elsewhere.} \end{cases} \tag{2.53}$$

The algorithm calculates an optimal filter $|H(e^{j\Omega})|$ for the desired frequency response. Optimal means that the maximum error between the calculated frequency response and the desired one is minimised (Cf. with Eq. (2.8) at Page 16). The

[10]The frequency response is described only in the interval $[-\pi; \pi]$.

resulting FIR filter should have odd symmetry, thus the phase of positive frequencies is $-90°$ and of negative $+90°$. The impulse responses will be calculated for 401 samples since the algorithm doesn't work for a length of 8192 samples:

$$|\tilde{H}[k]| := \begin{cases} 1 & 3 \leqq |k| \leqq 197 \\ 0 & k = 0 \text{ and } k = \pm 200 \\ \text{undefined} & \text{elsewhere,} \end{cases} \tag{2.54}$$

and

$$\angle\tilde{H}[k] := \begin{cases} +90 & -200 < k < 0 \\ -90 & 0 < k < 200 \\ 0 & \text{elsewhere.} \end{cases} \tag{2.55}$$

Later the Blackman window will also have a length of 401 samples to be able to compare both types of realization.

Real-world discrete frequency response of an Optimal FIR filter. The transfer function of a Hilbert transformer without special window sequence or Optimal FIR filter has been drawn in Figure 2.2 at Page 16. Since in practice only FFTs are being used the frequency response is discrete such as drawn in **Figure 2.9**, cf. with Definition 5. Only the first 20 frequency points $H[0] \cdots H[19]$ are shown due to the fact that it is better to present. The ripples are the same near $n = 200$ and at negative frequencies anyway.

With the Parks-Mc-Clellan algorithm the transfer function of Equations (2.54) and (2.55) looks like **Figure 2.10**. The slope is flatter than in Fig. 2.9 and the oscillations are much slower. Thus no frequency components of the signal exist which are boosted by the Hilbert transformer.

Simulation 1 — Simulation of the envelope error. In the following a simulation will be made to show the advantage of an Optimal FIR filter and later of a Blackman window. It will include ideal uniform sampling which will be discussed later. The input signal for the simulation is defined as a bandpass PAM[11] signal (Cf. with Appendix B) of a RFID system. The blockdiagram of our theoretic simulation is shown at **Figure 2.11** with following properties:

The ideal transmitter contains this setup:

- lowband $\leftrightarrow$ bandpass transformation: The carrier frequency $f_c := 13.56$ MHz.
- Length of a bit is $t_\mathrm{b} = \frac{32}{f_c} = 2.36\ \mu$s as defined in [ISO].
- According to [ISO08] sequences for Type A communication (PCD to PICC) should be used. The information {"Start of communication", "logical 0", "logical 0"} should be sent.

[11] In the real-world implementation Load Modulation is often used.

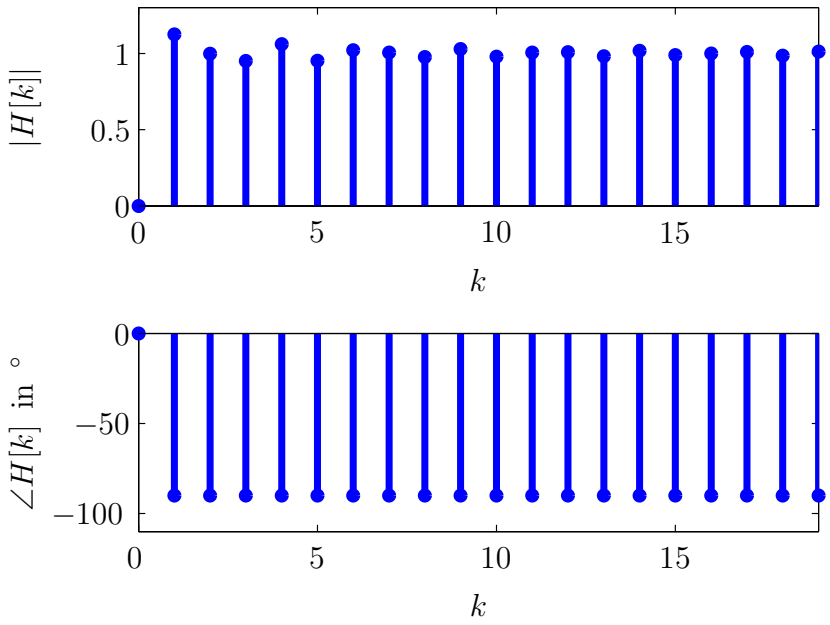


Figure 2.9: FFT of the impulse response of a Hilbert transformer ("rectangular window"). The order or length of the FIR filter is 400 samples.

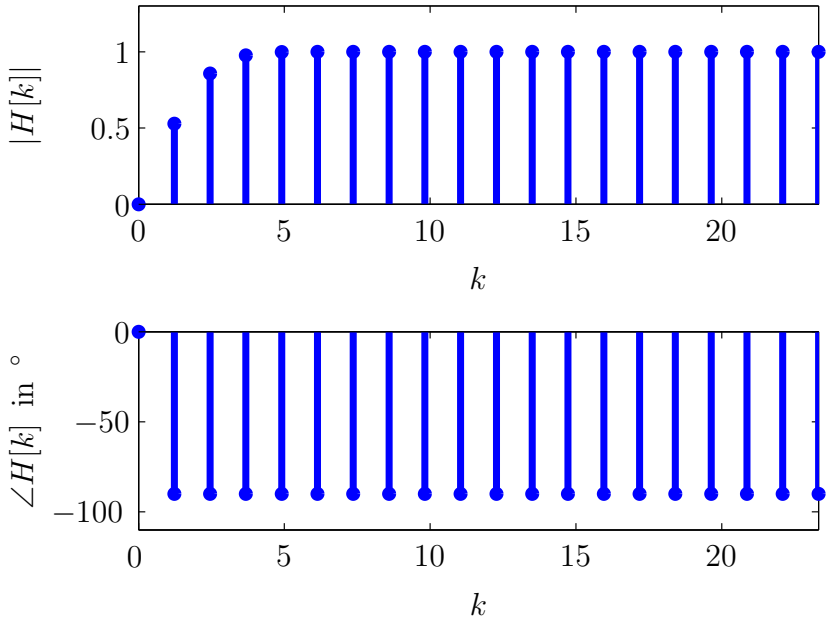


Figure 2.10: FFT of an Optimal FIR filter of a Hilbert transformer. The order or length of the FIR filter is 400 samples.

The ideal channel includes the whole path between transmitter and receiver. In our case it introduces no kind of distortion, i.e. the output of the LP↔BP transformation $u(t)$ isn't modified.

The ideal receiver which is simulated in Matlab®:

- Sampler: The bandpass signal is ideally sampled at the rate $f_{\mathrm{s,a}} := 200$ MS/s to get $u[n] = u(nT)$ with $T = 1/f_{\mathrm{s,a}}$.
- The output of the Envelope Detector $r[n] = |u_+[n]|$ and the spectrum of the Analytic Signal $\mathcal{F}\{u_+[n]\} = \mathcal{F}\{u[n] + \mathcal{H}\{u[n]\}\}$ will be analyzed.

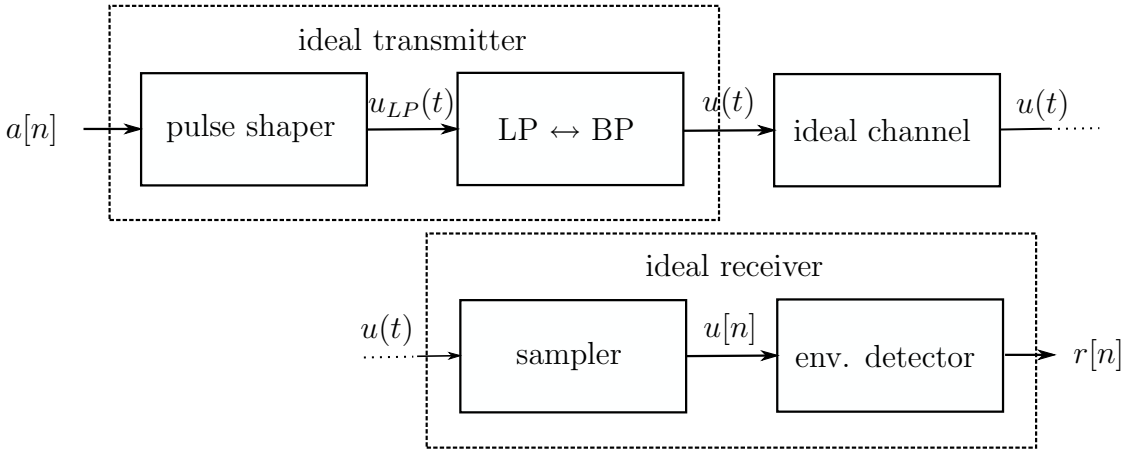


Figure 2.11: Block diagram of simulation.

The corresponding signal of {"Start of communication", "logical 0", "logical 0"} is according to [ISO08] $u_{\mathrm{LP}}(t) = g_{\mathrm{Z}}(t) + g_{\mathrm{X}}(t - T) + g_{\mathrm{Y}}(t - 2T)$ where

$$g_{\mathrm{X}}(t) = \mathrm{rect}\left(t - \frac{t_{\mathrm{b}}}{4}; \frac{t_{\mathrm{b}}}{4}\right) + \mathrm{rect}\left(t - \frac{t_{\mathrm{b}}}{2} - t_1 - \frac{t_{\mathrm{b}}/2 - t_1}{2}; \frac{t_{\mathrm{b}}/2 - t_1}{2}\right), \tag{2.56}$$

$$g_{\mathrm{Y}}(t) = \mathrm{rect}\left(t - \frac{t_{\mathrm{b}}}{2}; \frac{t_{\mathrm{b}}}{2}\right) \text{ and} \tag{2.57}$$

$$g_{\mathrm{Z}}(t) = \mathrm{rect}\left(t - \frac{t_{\mathrm{b}} - t_1}{2}; \frac{t_{\mathrm{b}} - t_1}{2}\right). \tag{2.58}$$

t_1 can be between $\frac{8}{f_c}$ and $\frac{10}{f_c}$. We are choosing $t_1 = \frac{8}{f_c}$. The obtained curve $u_{\mathrm{LP}}(t)$ which is sent[12] as baseband signal is plotted in **Figure 2.12**. After a lowpass ↔ bandpass transformation the obtained signal $u(t)$ is sent over an ideal channel

[12]It can also be viewed as the envelope of the bandpass signal.

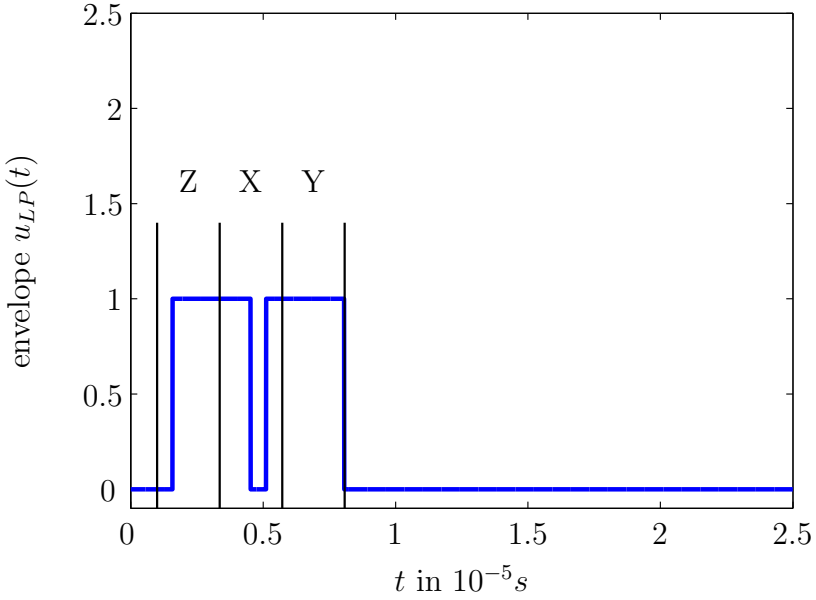


Figure 2.12: Baseband transmit signal. It has to be shifted into the bandpass signal and contains Sequences $\{\mathrm{Z}, \mathrm{X}, \mathrm{Y}\}$.

which doesn't modify the signal. The receiver consists of an ideal sampler and of our envelope detector. The second received symbol (Sequence Y in [ISO08]) is shown in **Figure 2.13** for a rectangular window and improvements like an Optimal FIR filter. One of the detected envelopes $r[n]$ is plotted in Figure 2.13 in relation to the original signal $u_{LP}(t)$, cf. with Appendix C. In **Figure 2.14** the rise in the second symbol (pulse shape X) and in **Figure 2.15** the fall in the same symbol is zoomed.

Windows and Optimal FIR filter. The next question is whether using a special window sequence like Blackman window or an Optimal FIR filter would reduce the error better. In our case the envelope detector is implemented in the time domain with Matlab®. Therefor it is possible to compare the three types: The rectangular window, the Blackman window and of course the Optimal FIR filter have a length of 401 samples. Thus the resulting Hilbert transformer also has 401 samples. The reason why 5000 samples hasn't been chosen is that it isn't possible to realize Optimal FIR filter of that length. A window with length of 5000 samples would have better results but the settling time of the FIR filter would be worse. To see the difference between the three implementations look at **Figure 2.16**.

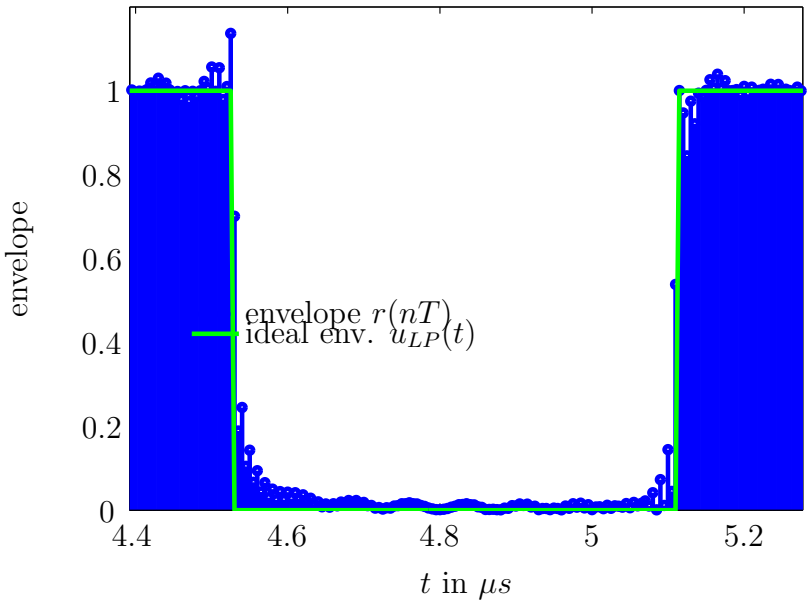


Figure 2.13: Ideal and detected envelopes of the second symbol (Sequence X). The graphic of the envelope $r(t)$ will be the same if a rectangular window, Optimal FIR filter or Blackman window is used since the difference is too small for this plot.

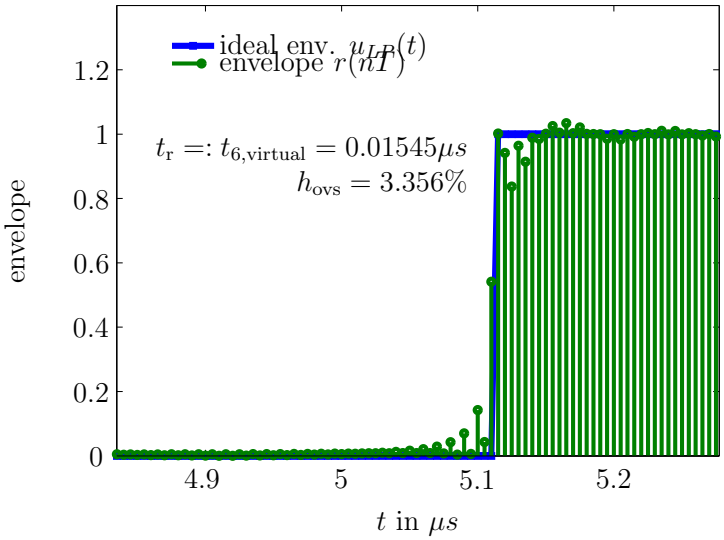


Figure 2.14: First jump between the amplitudes of 0 and 1 of ideal envelope and real-world envelope. The graphic of the envelope $r(t)$ is here the same for a rectangular window, Optimal FIR filter or Blackman window since the differences are too small for the plot! t_r specifies the rise time, and h_{ovs} the overshoot (cf. with Appendix C).

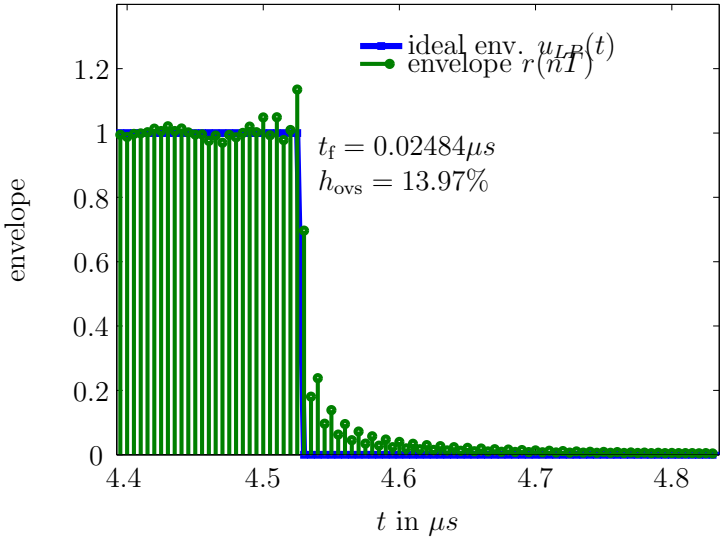


Figure 2.15: First fall between the amplitudes of 1 and 0 of ideal envelope and real-world envelope. The graphic of the envelope $r(t)$ is here the same for a rectangular window, Optimal FIR filter or Blackman window since the differences are too small for the plot! t_f specifies the fall time and h_ovs the overshoot (cf. with Appendix C).

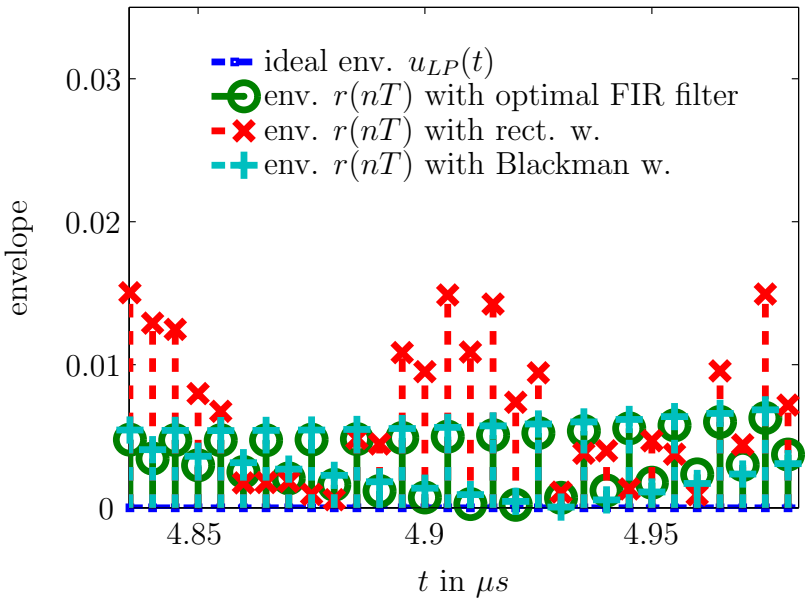


Figure 2.16: Difference between rectangular window, Blackman window and Optimal FIR filter. The center of the "pause" of the second symbol with pulse Sequence X is plotted. The pause is defined as the interval of the signal where it should be 0.

Errors of an Optimal FIR filter. We will analyze it further. **Figure 2.18** shows the difference in dB of the calculated envelopes:

$$\frac{\Delta}{\text{dB}} = 20\log_{10}(\frac{\text{detected envelope} - \text{ideal envelope}}{1}) \tag{2.59}$$

Δ_W specifies the difference using the rectangular window, Δ_O the Optimal FIR filter and Δ_B the Blackman window. One can distinguish these plots in three parts. The left and right parts are failures which are produced when the amplitude is 1. In the interval between 4.5 µs and 5.13 µs the signal should be zero.

Comparing both Figures 2.17 and 2.18 shows that indeed an Optimal FIR Hilbert filter is better. While the Optimal Filter is the best alternative with respect to Equation (2.8) other windows can be very good too.

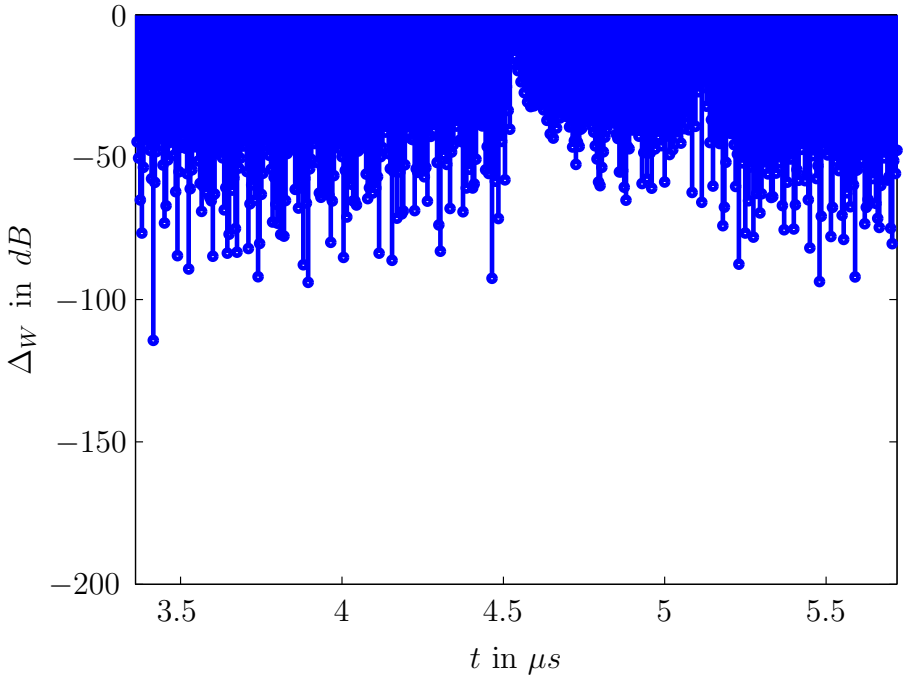


Figure 2.17: Difference (Error) between calculation of the envelope with rectangular window and ideal envelope.

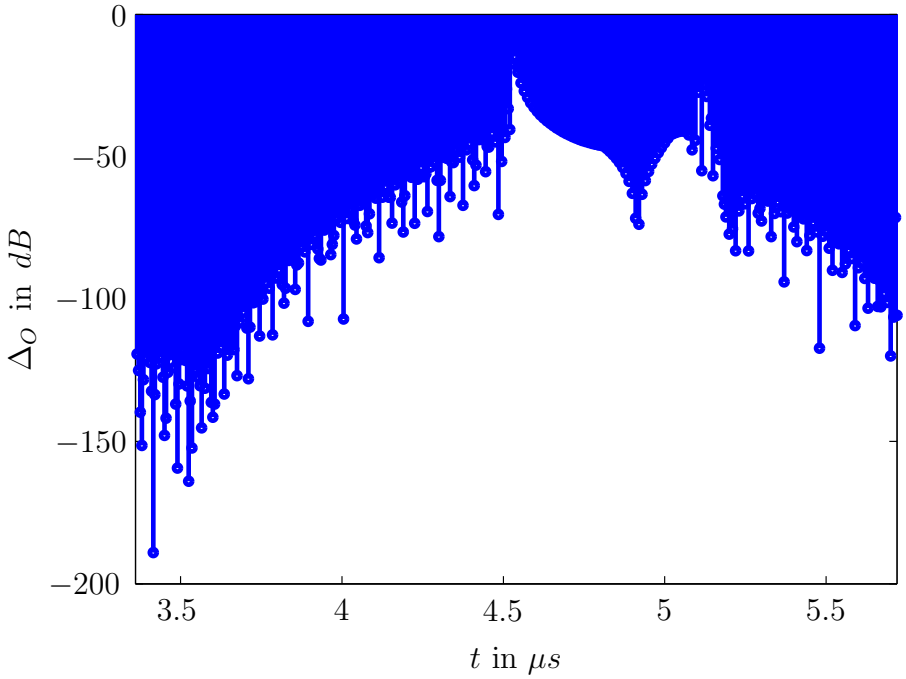


Figure 2.18: Difference (Error) between calculation of the envelope with Optimal filter and ideal envelope. Cf. with Figure 2.17 to see the improvement from the Optimal FIR filter.

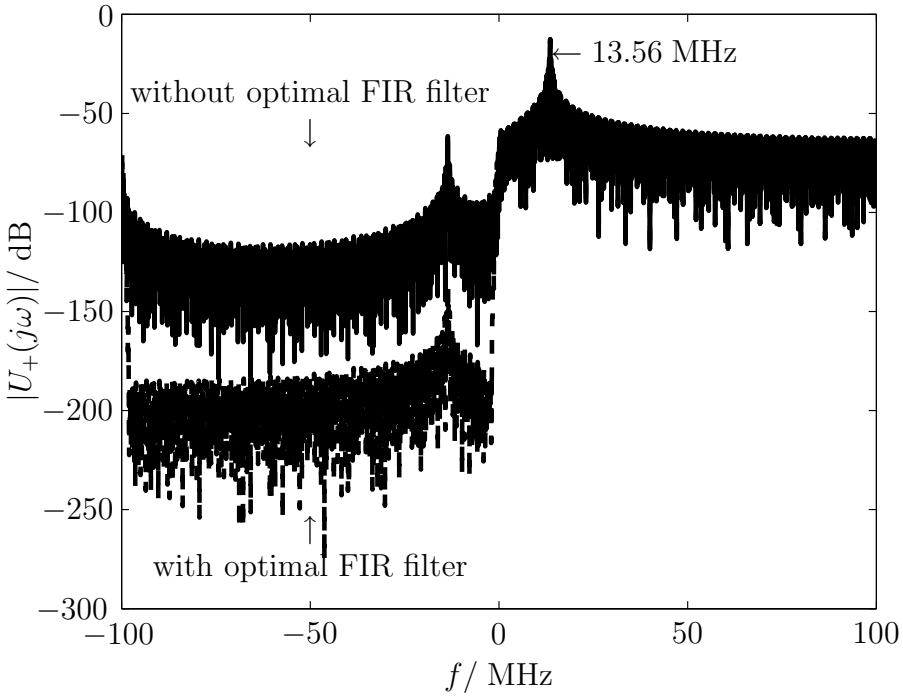


Figure 2.19: Spectrum of the analytic signal if an ideal bandpass filter existed. Components at negative frequencies aren't suppressed perfectly.

Spectrum of the Analytic Signal. To complete the analysis of the Optimal FIR filter the magnitude of the spectrum $|U_+(j\omega)|$ of the analytic signal (cf. with Equation (1.11) at Page 8) with and without Optimal FIR Hilbert transformer is plotted in **Figure 2.19**. The plot shows some interesting facts:

1. The signal components at negative frequencies aren't suppressed.
2. An Optimal FIR filter reduces these components (It can be shown that special windows sequences do the same).

Errors of a Blackman window. In the next **Figure 2.20** the same setup for an implementation with a Blackman window is used — cf. with Figure 2.16.

Conclusion. We have seen that indeed the rectangular window of the Fast Fourier Transform produces the worst result. But if one wants to decide which filter or window is used following items have to be defined first:

Time domain / frequency domain. Implementation in the time domain via a FIR filter or in the frequency domain using the FFT.

Max. length of the FIR filter or FFT. The Optimal FIR filter is only possible with "short" length, ie. the algorithm doesn't work for length of 5000 samples. Note that the length of a FIR filter corresponds to the delay of the filter.

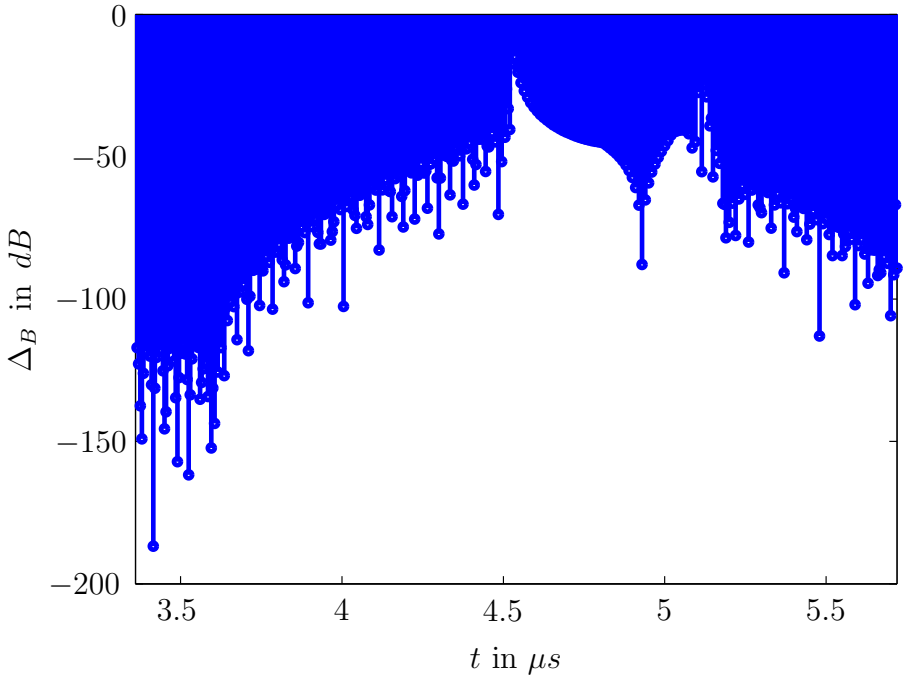


Figure 2.20: Errors between detected envelope using a Blackman window and ideal FIR filter. Cf. with Figures 2.16.

Bandwidth of the signal or pulse shape of the transmit filter. Since Hilbert transformers with different windows or Optimal FIR filters have different effective bandwiths the pulse shape is important. Effective bandwiths correspond to the rise of the frequency responses.

Note that we have used an Optimal FIR filter length of only 401 samples. If another type, length or bandwidth of the PAM signal would be used it isn't assured that the Blackman window would be better. Since the parameters of the Optimal FIR filter are variable it can be trimmed to very good results.

For the source code of Annex 7 of [ISO] the following constraint is important: Since the FFT of 5000 samples is applied the artfacts are smaller than in the previous simulation and can be even more reduced with a non-rectangular window.

2.6 Summary

The source code of [ISO] uses the FFT to implement the Hilbert transform which can be easily used for bandpass system — no bandpass $\leftrightarrow$ lowpass transform is necessary. But it can only calculate max. 5000 samples. Thus if one likes to improve it, one will have to use Overlap-add or Overlap-save. Another way would

be to use a FIR filter. Neither of them are generally better, the hardware decides the better variant.

We differentiated three kinds of errors:

- Since the source code should only give an idea how an envelope detector could be implemented there are some *errors in the source code*. These restrict the usage heavily. Only less than 5001 samples can be processed, but it has a 8192-points FFT. The big error is, that only 5000 samples are returned, so the rest is cut off. This leads to errors since the DFT (FFT) is periodic and some samples can be discarded. To avoid this the signal can be shifted, what has been done in the figures, but then the signal has to be shorter than 5000 samples. The source code uses many global variables, thus it would be very hard to use it in a project.

- Another part of the source code, in a separate file, is the *FFT routines*. The code is very inefficient and would lead to a slow implementation. The big problem is that it mixes `float` and `double` declarations of variables! Without them the result of the FFT and IFFT isn't that bad but could be improved with other kinds of implementation. Note that even with data types of `double` errors exist since multiplications and additions introduce them.

- One issue of the DFT is the so called Leakage. It results in a failure because using 5000 samples of a signal is like multiplying the signal with a rectangular window sequence (which is a sinc-function in the frequency domain). In the time domain it would be the same, the Hilbert transform FIR filter has to be cut off. A solution could be another window function or an Optimal FIR filter.

Part II

Influence of the analogue-to-digital conversion chain

In Part two we will discuss the developed envelope detector in a measurement system containing two models which have to be specified:

1. the source
2. the measurement path

The source should emulate the signal which is the input of the measurement path. It may be an oscilloscope in the simplest case.

Therefore the chapters are arranged like in following list:

Chapter 3 — Sampling and bandwidth of an oscilloscope — introduces some theory about sampling and oscilloscopes.

Chapter 4 — Models — describes a model for the signal source and of the measurement path.

Chapter 5 — Model implementation — goes into details and describes the implementation of the two models.

Chapter 6 — Common setup of source model — specifies parameters which will be used in Chapter 7.

Chapter 7 — Results of simulation and Conclusions — provides graphics and results of simulations.

3 Sampling and bandwidth of an oscilloscope

First we start with a short review of sampling which will be necessary in the following.

Definition 19 — Uniform sampling. Let $u(t)$ be a time continuous analogue signal. Uniform sampling is described by the relation

$$u[n] := u(nT) \; \forall n \in \mathbb{Z}. \tag{3.1}$$

Thus the sequence $\{u[n]\}$ consists of samples which are taken from the analogue signal every T seconds. T is called the sampling period or sampling interval and $f_s = 1/T$ the sampling frequency (rate) with the pseudo-unit $[f_s] = \mathrm{S/s}$.

One consequence is that the bandpass signal in Equation(1.22) can be sampled like the following equation:

$$u[n] \equiv u(nT) = r(nT)\cos\left(2\pi f_0 nT + \phi(nT)\right) = \\ r[n]\cos\left(2\pi\frac{f_0}{f_s}n + \phi[n]\right) =: r[n]\cos\left(\Omega n + \phi[n]\right). \tag{3.2}$$

$\Omega = 2\pi\frac{f_0}{f_s} \in [0; 2\pi]$ is a normalized frequency. The corresponding discrete-time Fourier transform was defined in Eq. (5).

Theorem 1 — Nyquist theorem. A continuous-time baseband[1] signal $u(t)$ which is bandlimited with highest frequency f_u can be uniquely recovered from its sample sequence $\{u[n]\}$ if the sampling frequency $f_s \geqq 2f_u$.

Aliasing. If the continuous-time signal is not bandlimited, ie. $f_u \geqq f_s/2$, then all frequencies $|f| > f_s/2$ will appear in the interval $[0; 2\pi]$ of the normalized frequency of the sampled signal. Reconstructing of the original signal won't be possible since sampling won't be bijective anymore. This unwanted effect is called aliasing.

[1] A continuous-time bandpass signal $u(t)$ with a bandwidth B can be uniquely recovered if $f_s \geqq \frac{1}{B}$. Note that the bandwith of our signal won't be limited.

In some applications, like in the environment of RFID, the bandwidth of a signal isn't restricted. The whole measurement (f.e. an oscilloscope) acts like a low- or bandpass filter. So we have to discuss them in the next section.

3.1 Sampling techniques

There are three types of sampling techniques:

Real-time sampling technique. The uniform sampling discussed above is — if implemented in the real-world — the so called real-time or single-shot technique. All samples are taken from a single trigger.

The advantage of this technique is that repetitive and non-repetitive signals can be sampled. T has to be as exact as possible.

Equivalent-time sampling technique. The signal is assumed to be repetitive and it can be sampled more than once with a time shift between the periods. Thus more than one trigger has to be used. By way of example we could have a periodic signal $u(t)$ and two triggers. The first one samples the first period and gets $\{u[0], u[2], u[4], u[6]\}$. The second trigger samples the second period to obtain $\{u[1], u[3], u[5], u[7]\}$. Therefore the waveform is built up after the second step to $\{u[0], u[1], u[2], u[3], u[4], u[5], u[6], u[7]\}$.

The big advantage is that the effective sampling rate T can be achieved with triggers of period time $2T$, hence the resolution $f_{s,e}$ can be much higher than the one from the real-time sampling:

$$f_{s,e} = nf_s \; \forall n \in \mathbb{N} \tag{3.3}$$

But it is also a source of errors since the triggers have to be separated exactly by T/n seconds.

Sequential sampling technique. Sequential sampling is an extreme case of equivalent sampling, where each trigger takes only one sample of the period. This is the final type of sampling and allows sample frequencies $f_s > 20$ GS/s.

RFID. In the field of RFID repetitive or periodic signals cannot be assumed, so one has to use the real-time technique.

3.2 Bandwidth of frontend hardware

Lowpass filter. Beside the sampling rate the bandwidth of the frontend is an important property of the system. The ideal transfer function of the frontend

hardware can be assumed to be a lowpass filter $H_h(j\omega)$ which is defined by

$$H_h(j\omega) := \begin{cases} C & |\omega| \leqq \omega_c \\ 0 & |\omega| > \omega_c \end{cases} \tag{3.4}$$

where $\omega_c = 2\pi f_c$ is the cut-off angular frequency and $C \leqq 0$ dB a constant. Note that in the real-world the fall is finite and so the cut-off angular frequency is defined by

$$\omega_{-3\text{ dB}} := \omega|_{H_h(j\omega)=C-3\text{ dB}} = 2\pi f_{-3\text{ dB}}. \tag{3.5}$$

In the case of an oscilloscope $f_{-3\text{ dB}}$ of the probe would be important since — in a theoretically point of view — the sampling frequency

$$f_s \overset{!}{>} 2\min\{f_c, f_u\}. \tag{3.6}$$

Bandwidth of probe and scope. Oscilloscopes with bandwidth specifications $f_{-3\text{ dB}} < 1$ GHz have the same frequency response like a Gaussian filter. According to [Rap02] the Gaussian filter is defined by

$$h_{\text{LP,G}}(t, \alpha) := \frac{\sqrt{\pi}}{\alpha} e^{-\left(\frac{\pi t}{\alpha}\right)^2} \tag{3.7}$$

with the frequency response

$$H_{\text{LP,G}}(j\omega, \alpha) = e^{-\left(\frac{\alpha\omega}{2\pi}\right)^2} \tag{3.8}$$

and

$$\alpha = \frac{0.5887}{f_{-3\text{ dB}}}. \tag{3.9}$$

If the probe has the same property then the cut-off frequency can be easily calculated with

$$f_{-3\text{ dB}} = \frac{1}{\left(\frac{1}{f_{-3\text{ dB,probe}}}\right)^2 + \left(\frac{1}{f_{-3\text{ dB,scope}}}\right)^2}. \tag{3.10}$$

If the oscilloscope has a bandwidth with $f_{-3\text{ dB}} \gtrsim 1$ GHz the frequency response will be — the so called — *flat frequency response*. Below $f_{-3\text{ dB}}$ the response is flatter than that from Gaussian filter and the roll-off near the point where the magnitude has reduced to -3 dB is sharp-edged. On the other side the disadvantage exists that one cannot calculate the whole bandwidth of the measurement chain and has to ask the vendor. For the sake of simplicity we will use a Gaussian frequency response in the following and note that the systems with flat frequency response are generally better.

Required bandwidth. According to [agia] we can introduce the "knee" frequency f_{knee} of a signal which is defined as the "maximum practical frequency component". It is defined as

$$f_{\text{knee}} := \frac{0.5}{t_r} \tag{3.11}$$

where t_r is the rise time from 10% to 90% of the signal. To calculate the required bandwidth $f_{-3\,\text{dB}}$ of the oscilloscope (including the probe) **Table 3.1** can be used.

required accuracy	Gaussian response	maximally-flat response
20%	$f_{-3\,\text{dB}} = f_{\text{knee}}$	$f_{-3\,\text{dB}} = f_{\text{knee}}$
10%	$f_{-3\,\text{dB}} = 1.3 f_{\text{knee}}$	$f_{-3\,\text{dB}} = 1.2 f_{\text{knee}}$
3%	$f_{-3\,\text{dB}} = 1.9 f_{\text{knee}}$	$f_{-3\,\text{dB}} = 1.4 f_{\text{knee}}$

Table 3.1: Calculation of the bandwidth $f_{-3\,\text{dB}}$ of a signal.

Required bandwidth of the signal The paper [ISO08] defines the maximum rise time (worst case) $t_r := 0.442$ µs. Since in the following we are analyzing the band-pass system $t_r := 12$ ns. With Equation (3.11) and Table 3.1 we get the values provided in **Table 3.2**. Note that nowadays all oscilloscopes have bandwiths $f_{-3\,\text{dB}} > 100$ MHz.

required accuracy	$f_{-3\,\text{dB}}$ of Gaussian response	$f_{-3\,\text{dB}}$ of maximally-flat response
20%	41.7 MHz	41.7 MHz
10%	54.2 MHz	50 MHz
3%	79.2 MHz	58.3 MHz

Table 3.2: Calculated values of the bandwidth $f_{-3\,\text{dB}}$ of the signal defined in [ISO].

Minimum sampling frequency. According to [agia] a conservative calculation of the minimum sampling frequency which is necessary can be calculated with

$$f_{\text{s,min}} = 4 f_{-3\,\text{dB}}. \tag{3.12}$$

3.3 Real-world sampling

Aperture time. Equation (3.1) can also be written as

$$u[n] = u(nT) = \int_{-\infty}^{\infty} u(t)\delta(t - nT)dt \;\forall n \in \mathbb{Z}. \tag{3.13}$$

In the real-world case a Dirac impulse $\delta(t)$ isn't possible, therefore it has to be rewritten using a weight-function

$$w(t) = C(t)\,\mathrm{rect}\left(\frac{t - T_{ap}}{2}; T_{ap}\right) = \begin{cases} C(t) & -T_{ap}/2 \leqq t \leqq T_{ap}/2 \\ 0 & \text{elsewhere} \end{cases} \tag{3.14}$$

where T_{ap} is the duration of the aperture which is called aperture time. $C(t)$ is a function of time. Equation (3.1) enhances to

$$u[n] = \int_{-\infty}^{-\infty} u(t)w(t - nT)dt \;\forall n \in \mathbb{Z}. \tag{3.15}$$

The transfer function of $w(t)$ has a $\sin(x)/x$ shape which has a lowpass characteristic and therefore introduces errors.

Aperture jitter. Another problem is the random aperture jitter ε_T in seconds[2]. It is an uncertainty of the sample time $nT + \varepsilon_T$ whereby wrong amplitudes are sampled and the whole error increases with increasing sampling frequency.

Real-world sampling Both errors lead to a better mathematican description of the sampling:

$$u'[n] := u(nT + \varepsilon_T) = \int_{-\infty}^{\infty} u(t)w(t - nT + \varepsilon_T)dt = \int_{-\frac{T_{ap}}{2} - nT + \varepsilon_T}^{\frac{T_{ap}}{2} - nT + \varepsilon_T} C(t)u(t)dt. \tag{3.16}$$

Note that ε_T is random and the local oscillator (LO) defines the pdf f_{ε_T}.

[2]With an abuse of notation upper-case letters don't specify random variables in this section.

4 Models

4.1 Envelope Detectors applied to real-world signals

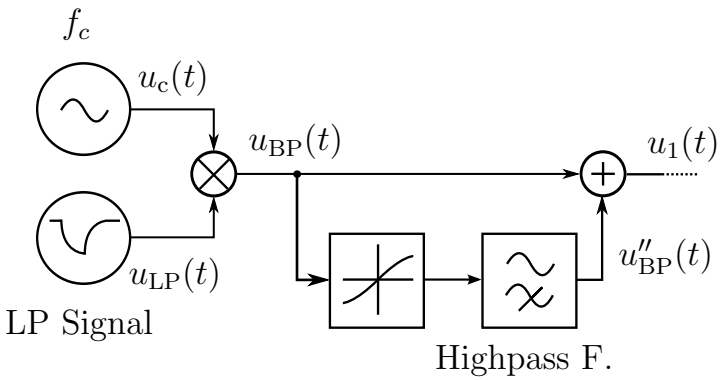


Figure 4.1: Source generation.

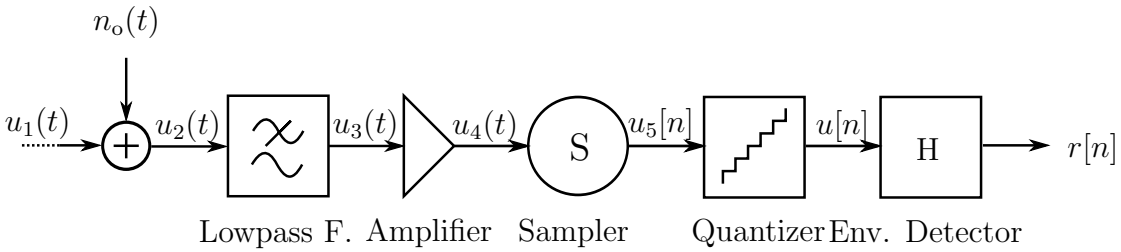


Figure 4.2: Measurement chain.

In our previous discussions we have observed that the implementation of an envelope detector has many aspects and many errors can be introduced. The simulations have used ideal samplers and ideal quantization has been assumed. This chapter will discuss the whole conversion chain from analogue to digital signals before an envelope detector. This could be for example an oscilloscope, where the chain includes a prefilter (probe), an amplifier, a sampler and a quantizer. It will include amongst others aliasing effects, the sampling rate, the bandwidth of the whole chain, the quantization error and noise.

4.1.1 Source generation

Oscillator. The transmitter is plotted in **Figure 4.1**. The source produces a complex-valued normalized signal with an in-phase and a quadrature component[1]:

$$u_{\mathrm{c}}(t) = e^{j\omega_c t} = u_{\mathrm{c,I}}(t) + j u_{\mathrm{c,Q}}(t), \tag{4.1}$$

$$[u_{\mathrm{c,I}}(t)] = [u_{\mathrm{c,Q}}(t)] = 1, \tag{4.2}$$

ie. the signal is normalized and thus proportional to voltage. We write $u_{\mathrm{c}}(t)$, but note that an analogue signal can only be simulated using a discrete-time signal with very high sampling rate!

Phase noise. Since we want to emulate a real-world oscillator, phase noise $N_{\mathrm{signal}}(t)$ is added to $u_{\mathrm{c}}(t)$. According to [SMMW00] the noise source in electrical circuits can be split into two groups.

1. Device noise includes thermal, shot and flicker noise.
2. Interference includes substrate and supply noise.

Noise due to an oscillator. Since the signal is generated using an oscillator phase noise is a very important property. The output on an idea oscillator would be

$$U(t) = A\cos(\omega_0 t + \phi) \tag{4.3}$$

while in reality amplitude A and phase ϕ are functions of time written as

$$U(t) = A(t)\cos(\omega_0 t + \phi(t)) = A(t)\cos\left((\omega_0 + \Delta\omega(t))t\right). \tag{4.4}$$

One consequence is that instead of two spectral components $\pm\omega_0$ the signal has sidebands. One characteristic is the *single sideband noise spectral density* defined[2] by

$$L(\Delta\omega) := L_{\mathrm{phase}}(\Delta\omega) + L_{\mathrm{amplitude}}(\Delta\omega) = 10 \cdot \log\left(\frac{P_{\mathrm{sideband}}(\omega_0 + \Delta\omega)}{P_{\mathrm{carrier}}}\right), \tag{4.5}$$

$$[L] = 1 \text{ dBc/Hz}. \tag{4.6}$$

$P_{\mathrm{sideband}}(\omega_0 + \Delta\omega)$ is the single sideband power at a frequency offset of $\Delta\omega$ in respect to the carrier frequency ω_0 with a bandwidth of 1 Hz. It can be shown (cf. [SMMW00]) that L_{phase} contains 3 different parts with constants c', c'', c''':

$$L_{\mathrm{phase}}(\Delta\omega) = \underbrace{c'}_{\text{temperature}} + \underbrace{\frac{c''}{\Delta\omega^2}}_{\text{quality of loading}} + \underbrace{\frac{c'''}{\Delta\omega^3}}_{\text{flicker noise}}. \tag{4.7}$$

The flicker noise is defined by the quality of the crystal; the $(\Delta\omega)^2$ term by the circuit.

[1] This is done to be able to enhance the simulation easily.

[2] $\frac{P}{1\text{ dBc}} := 10 \cdot \log\left(\frac{P}{P_{\mathrm{carrier}}}\right)$

Modulation. Then the data stream modulates the I-part of the signal, while the Q-part $u_{\mathrm{c,Q}}(t) := 0$. This means that we are using ASK. This makes it possible to simulate phase shifts of the complex signal. It would reduce the energy of the I-component and would increase the energy of the Q-component.

The lowpass signal $u_{\mathrm{LP}}(t)$ is generated by a signal shaper which generates the signal specified in [ISO08]. Since this would introduce some errors caused by the Gibbs phenomenon owing to points of discontinuity the signal has to be smoothed. This is done by a lowpass filter.

Non-linearity system. There is a branch which attenuates the signal $u_{\mathrm{BP}}(t)$ and adds non-linearity signal components owing to SHOCKLEY diodes. The characteristic curve of this type of diode is defined by

$$I_{\mathrm{D}} = I_{\mathrm{S}} \left(e^{\frac{U_D}{n\,U_T}} - 1 \right) \tag{4.8}$$

where U_{D} is the voltage across the diode, I_{D} the diode current, U_{T} is the thermal voltage, I_{S} the saturation current and n the emission coefficient. One will expect that a characteristic curve of the input-output-relation of the block is proportional to $e^x - 1$ if $x := \frac{U_D}{n\,U_T}$. Thus the third-order Taylor series approximation

$$e^x - 1 = \sum_{n=0}^{\infty} \frac{x^n}{n!} - 1 = x + x^2/2 + x^3/6 + O\{x^4\} \tag{4.9}$$

can be used. If we substitute x by a function or signal $x(t) = \sin(2\pi f t)$ then Eq. (4.9) would produce new frequency components at $2f$, $3f$, etc. This could be easily verified by calculating of $\sin(2\pi f t) + (\sin(2\pi f t))^2/2 + (\sin(2\pi f t))^3/6$ using addition theorems.

Since in our simulation we want to be able to define the signal strength of each frequency component separately, the characteristic curve can be generalized and written by

$$c'x + c''x^2 + c''x^3 + O\{x^4\} \qquad\qquad c', c'', c''' \in \mathbb{R}. \tag{4.10}$$

Now the input-output-relation is

$$u'_{\mathrm{BP}}(t) = u'_{\mathrm{BP,I}}(t) + j u'_{\mathrm{BP,Q}}(t) \text{ with} \tag{4.11}$$

$$u'_{\mathrm{BP,I}}(t) = c' u_{\mathrm{BP,I}}(t) + c'' u^2_{\mathrm{BP,I}}(t) + c''' u^2_{\mathrm{BP,I}}(t) \text{ and} \tag{4.12}$$

$$u'_{\mathrm{BP,Q}}(t) = c' u_{\mathrm{BP,Q}}(t) + c'' u^2_{\mathrm{BP,Q}}(t) + c''' u^2_{\mathrm{BP,Q}}(t). \tag{4.13}$$

Note that we have splitted the complex signal into a real and a imaginary part to apply the equation separately. The constants c',c'' and c''' are our three parameters of the block which can be changed for every specific simulation.

Highpass filter. A highpass filter is required since $u'_{\mathrm{BP}}(t)$ has many frequency components close to 0 Hz which shouldn't exist.

$$u''_{\mathrm{BP}}(t) = H_{\mathrm{LP}} \star u'_{\mathrm{BP}}(t) \tag{4.14}$$

where H_{LP} is the frequency response of the lowpass filter.

Summation – complex-valued to real-valued signal. Later $u''_{\mathrm{BP}}(t)$ is added to $u_{\mathrm{BP}}(t)$ to get the corrupted signal. Since we can only transmit real-valued signals we have to translate it using a function $f : \mathbb{C} \to \mathbb{R}$:

$$\begin{aligned} u_1(t) =& f\left(u''_{\mathrm{BP}}(t) + u_{\mathrm{BP}}(t)\right) && (4.15)\\ =& \left|u''_{\mathrm{BP}}(t) + u_{\mathrm{BP}}(t)\right| \cos\left(\phi(t)\right), && (4.16)\\ \phi(t) =& \tan^{-1}\left(\frac{\mathrm{Im}\left(u''_{\mathrm{BP}}(t) + u_{\mathrm{BP}}(t)\right)}{\mathrm{Re}\left(u''_{\mathrm{BP}}(t) + u_{\mathrm{BP}}(t)\right)}\right) && (4.17) \end{aligned}$$

4.1.2 Measurement chain

Noise. The whole noise $N(t)$ contains a sum of different types: The noise from *a*) the received signal (phase and temperature noise), *b*) the probe, *c*) the attenuation, *d*) the amplifier *e*) the sampler and *f*) from the quantization.

$$\begin{aligned} N(t) =& N_{\mathrm{signal}}(t) + N_{\mathrm{probe}}(t) + N_{\mathrm{amplifier}}(t) + && (4.18)\\ & N_{\mathrm{attenuation}}(t) + N_{\mathrm{sampler}}(t) + N_{\mathrm{quantizer}}(t). && (4.19) \end{aligned}$$

The noise of the received signal $N_{\mathrm{signal}}(t)$ of an proximity coupling device — PCD — contains phase noise from the oscillator. In a datasheet of an oscilloscope often only the whole noise is specified. Therefore we can only use one function $N_{\mathrm{o}}(t)$ and assume due to the central limit theorem additive white Gaussian noise (AWGN) with zero mean and variance $\sigma^2_{\mathrm{N_o}}$. In the end we get two noise components:

$$N(t) = N_{\mathrm{signal}}(t) + N_{\mathrm{o}}(t). \tag{4.20}$$

Additive white Gaussian noise. Additive white Gaussian noise $N_{\mathrm{o}}(t)$ is identified as the noise from the digital sampling oscilloscope and will be added:

$$u_2(t) = u_1(t) + N_{\mathrm{o}}(t). \tag{4.21}$$

Lowpass filter. **Figure 4.2** contains all blocks from the measurement path (oscilloscope). The first block contains the lowpass filter which has a specified cut-off frequency[3] $f_{-3\,\mathrm{dB}}$. It is the combined frequency response of probe and oscilloscope.

[3]Here it can be also called "bandwidth of the oscilloscope".

In the common case we can assume that the filter is a Gaussian filter. There are oscilloscopes with steeper but non-specified slopes. It follows

$$u_3(t) = u_2(t) \star H_{\mathrm{LP,G}} \tag{4.22}$$

using Definition (4.2).

Non-linear elements. The following non-linear block contains three blocks of an oscilloscope — the attenuation, the amplifier and the non-linearities of the quantizer.

We are using a very common model. The cubic model is described in [Che06] where the input-output-relation is defined by

$$u_4(t) = au_3(t) - bu_3^2(t) - cu_3^3(t). \tag{4.23}$$

$a = G$ is the linear gain. For the sake of simplicity the coefficient $b := 0$ since it doesn't produce intermodulation products close to the frequencies of the signal and is assumed that they are filtered out. A common parameter describing an amplifier is the third order input intercept point IIP3.

Let the input signal of an amplifier contain two tones

$$u_3(t) := A\left[\cos(\omega_1 t) + \cos(\omega_2 t)\right]. \tag{4.24}$$

Inserting it into Equation (4.23) and applying some trigonometric identities gives

$$\begin{aligned} u_4(t) &= aA\left[\cos(\omega_1 t) + \cos(\omega_2 t)\right] \\ &= bA^2 + b\frac{A^2}{2}\left[\cos(2\omega_1 t) + \cos(2\omega_2 t)\right] \\ &- bA^2\left[\cos(\omega_1 t + \omega_2 t) + \cos(\omega_1 t - \omega_2 t)\right] \\ &- c\frac{9}{4}A^3\left[\cos(\omega_1 t) + \cos(\omega_2 t)\right] \\ &- c\frac{1}{4}A^3\left[\cos(3\omega_1 t) + \cos(3\omega_2 t)\right] \\ &- c\frac{3}{4}A^3\left[\cos(2\omega_1 t + \omega_2 t) + \cos(2\omega_2 t + \omega 1_t)\right] \\ &- c\frac{3}{4}A^3\left[\cos(2\omega_1 t - \omega_2 t) + \cos(2\omega_2 t - \omega 1_t)\right] \end{aligned} \tag{4.25}$$

The terms involving $2\omega_1 - \omega_2$ and $2\omega_2 - \omega_1$ are components of third order which are close to ω_1 and ω_2 and are therefore important. The IIP3 is the input amplitude where components of first and third order are equal:

$$aA \equiv \frac{3}{4}cA^3 \tag{4.26}$$

Thus

$$c = \frac{4a}{3A^2}. \tag{4.27}$$

Note that

$$\begin{aligned} \text{IIP3} &:= 10 \log_{10} \left[A^2 \frac{1000}{2R} \right], \\ [\text{IIP3}] &= \text{dBm}, \\ [R] &= \Omega, \\ [A] &= \text{V}, \end{aligned} \tag{4.28}$$

where R is the reference resistance[4]. Here $[\,\cdot\,]$ specifies the unit of the quantity[5].

Sampler. Since the time-continuous signal $u_4(t)$ can only be simulated as discrete-time signal, the sampling rate $f_{\mathrm{s,a}} = 1/T_{\mathrm{s,a}}$ of the pseudo-analogue signal has to be very high. Therefore in this case the sampler is transrating to a sampling rate of $f_{\mathrm{s}} = 1/T$. Note that $f_{\mathrm{s,a}} \gg f_{\mathrm{s}}$. Transrating, which is also called re-sampling, is the same as the conversion of a discrete signal to a analog signal, which then will be sampled with the new sampling rate. Thus aliasing can occur.

Thus we have to substitute t by $mT_{\mathrm{s,a}}$

$$u_4(t) = u_4(mT_{\mathrm{s,a}}) =: u_4[m] \tag{4.29}$$

The reconstruction of an analog signal from a sampled signal is

$$u_{\mathrm{a}}(t) = \sum_{m=-\infty}^{\infty} u_4[m] g(t - mT_{\mathrm{s,a}}) \tag{4.30}$$

with[6]

$$g(t) = \operatorname{sinc}\left(\frac{\pi}{T_{\mathrm{s,a}}} t \right). \tag{4.31}$$

$g(t)$ is called the interpolation function which avoids aliasing (= rectangle in the frequency domain). If this signal is sampled again with $f_{\mathrm{s}} = 1/T$ we will get the following equation:

$$u_5[n] := u_{\mathrm{a}}(mT) = \sum_{m=-\infty}^{\infty} u_4[m] g(nT - mT_{\mathrm{s,a}}) \tag{4.32}$$

[4]The factor 2 relates peak value A to the root mean square value (RMS).

[5]A physical quantity X can be expressed as product of numerical value $\{X\}$ and physical unit $[X]$: $X = \{X\}\,[X]$

[6]$\operatorname{sinc}(x) := \sin(x)/x$ with $\operatorname{sinc}(0) := 1$.

In the following we will simplify the process and assume that $R = T/T_{\mathrm{s,a}} = L/M \in \mathbb{Q}$ (R is a rational factor). It can be shown that this leads us to

$$u_5[n] = \sum_{m=-\infty}^{\infty} u_4[m] g(nM - mL) = \sum_{m=-\infty}^{\infty} u_4[m] \mathrm{sinc}\left(\theta_g (nM - mL)\right) \tag{4.33}$$

with $\theta_g = \min\left(\frac{\pi}{L}, \frac{\pi}{M}\right)$.

In our case we want to simulate only the sampler without an anti-aliasing filter with a re-sampler. Therefore aliasing is possible. We also don't need interpolation, since $f_{\mathrm{s,a}} \gg f_{\mathrm{s}}$ or $T \gg T_{\mathrm{s,a}}$. This leads us with $g(t) = \mathcal{F}^{-1}\{1\} = \delta(t)$ to the very simple input-output relation:

$$u_5[n] := u_4[mR] = u_4(mT_{\mathrm{s,a}}R). \tag{4.34}$$

The sampler introduces noise caused by the time jitter of the ADC. The reason of the jitter is the non-ideal oscillator of the sampler. Oscilloscopes have very accurate oscillators included. An Agilent oscilloscope from the 3000 series has a jitter about 15 ps. The effect is time jitter of the trigger and amplitudes of the sampled signal have failures, denoted by $N_{\mathrm{sampler}}(t)$. We will assume that due to the central limit theorem the distribution of the time jitter can be assumed as Gaussian.

There is also a short approximation of the amplitude error due to the time jitter. Since we sample a bandpass signal the highest rise of the signal (= derivation of the signal) can be calculated with a sinus. In the non-modulated case we have at 13.56 MHz the maximum rise:

$$k_{\max} = \cos(2\pi f t) 2\pi f|_{f=13.56\ \mathrm{MHz}, t=0\ \mathrm{s}} = 85.2\ \mathrm{MHz} \tag{4.35}$$

In our simulation the pseudo sample period of the analogue signal $T_{\mathrm{s,a}} = 1/f_{\mathrm{s,a}} = 1/13.56\ \mathrm{GHz} = 73.746\ \mathrm{ps}$. Thus the amplitude failure would be 0.314% of the amplitude by a time jitter of $T_{\mathrm{s,a}}/2 = 36.87$ ps. If the Agilent oscilloscope (3000 series) was used then the amplitude error would be 0.13%.

Quantizer. Since 8 bit analog-digital-converter are very common an 8 bit quantizer is used by default. It can be improved to 12 bit or 16 bit. We will write these sets of numbers $\mathbb{I}_k = \{0 \cdots 2^k - 1\}$ with the number k of bits. An interesting paper which collects the history and practice of the quantization can be found in [GN91].

Since the signal is assumed to be symmetric around zero, the *quantization interval* of a k-bit ADC with the amplitude limits of the input signal $-a/2$ and $a/2$ is

$$q = a/2^{k-1}. \tag{4.36}$$

Then the output of the quantizer block is

$$u[n] = \begin{cases} -a/2 & u_5[n] \leqq -a/2 \\ [u_5[n]/q]\, q & -a/2 < u_5[n] < a/2 \\ a/2 & u_5[n] \geqq a/2 \end{cases} \tag{4.37}$$

$[\,\cdot\,]$ denotes the round function. Note that small errors occur since an injective mapping $\tilde{\tau} : \mathbb{I}_k \rightarrow \mathbb{Q}_{64}$ is automaticly done. The new signal is $u[n] \in \mathbb{Q}_{64}$. It is the input of the following.

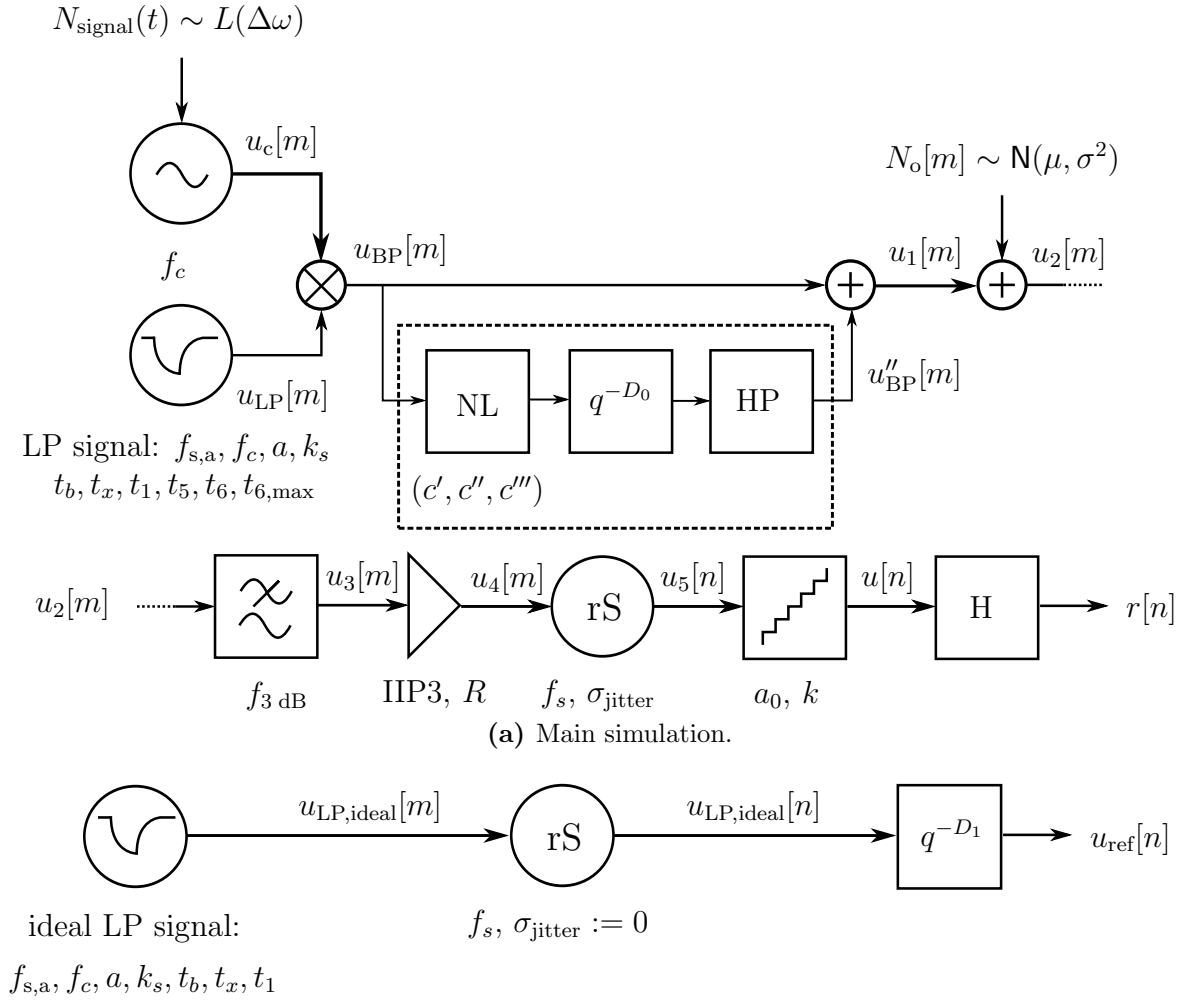
Envelope detector. The well-known Hilbert envelope detector (H) is the next element in the chain where its output is the detected envelope $r[n]$. Note that the envelope detector uses a 64 bit double precision floating point data type with the set of numbers $\mathbb{Q}_{64}$ instead of a fixed point data type which is used as output from the quantizer:

$$r[n] = |\mathcal{A}\{u[n]\}| \in \mathbb{Q}_{64}. \tag{4.38}$$

Going into details and using Equation (1.31) on Page 11 gives one possibility of an implementation:

$$r[n] = |\mathcal{F}^{-1}\{\mathcal{F}\{u[n]\} \cdot \mathcal{F}\{h[n]\}\}|. \tag{4.39}$$

5 Model implementation



(a) Main simulation.

(b) Reference path.

Figure 5.1: Blockdiagram of the testbed. NL is a non-linear system, HP is a highpass filter, rS denotes re-sampling and H the Hilbert envelope detector. The variables below the blocks specify the parameters which can be easily changed for the simulation.

This chapter will enhance the models provided in the previous Chapter 4 to get a mathematical description which can be implemented in Matlab® or another script language. Thus the time-continuous part will be formulated in the time-discrete

domain and a reference path will be added to calculate a quantity called error. The overall system is shown in **Figures 5.1** and will be discussed in the following.

5.1 Source model

The blockdiagram of the source model is plotted in **Figure 5.1a**.

LP signal. The signal is generated by two steps. First a "Type A" signal $u'_{\mathrm{LP}}[m]$ is shaped according to the RFID standard [ISO08]. The time parameters t_b, t_x, t_1, t_5, t_6 and $t_{6,\max}$ from there are all given in periods (cycles) of carrier, thus $[t] = 1$. The resulting signal

$$u'_{\mathrm{LP}}[m] = \begin{cases} 0 & m < 0 \text{ and } m > M \\ A[m] & 0 \leqq m \leqq M \end{cases} \tag{5.1}$$

with normalized amplitudes $A[m]$ at time indices m. M is an arbitrary time index which we define sloppily as "end of the signal"; whereas we define $m = 0$ as "start of the signal".

One disadvantage is the existence of points of discontinuity in the pulse shape due to the definitions in [ISO08]. To get a better model of a real-world signal it has to be smoothed. This is done by an N-order Gaussian FIR filter with a bandwidth of $f_{-3\,\mathrm{dB}} := 10^7$ Hz and frequency response $H_{\mathrm{LP,G}}$:

$$u_{\mathrm{LP}}[m] = u'_{\mathrm{LP}}[m] \star H_{\mathrm{LP,G}} \tag{5.2}$$

The lowpass filter introduces a delay of

$$D = \begin{cases} \frac{N-1}{2} & N \text{ odd} \\ \frac{N}{2} & N \text{ even} \end{cases} \tag{5.3}$$

samples. The samples of time index $m \leqq D$ are transient oscillations and therefore can be set to zero. But the signal has also transient osciallations after D samples, which will be taken into account later.

Bandpass signal. The carrier signal is defined by

$$u'_c[m] := \cos\left(2\pi \frac{f_c}{f_{s,a}} m + \Phi[m]\right). \tag{5.4}$$

We assume small phase noise. According to Equation (4.6) at Page 51 we define the shape of $L(\Delta\omega)$ in the spectrum with frequency-amplitude tuples:

$$(\Delta\omega, L) \tag{5.5}$$

From these tuples the discrete single sideband noise spectral density $L[\Delta m]$ can be calculated approximately. Applying it to Equation 5.4 the generated carrier signal is written as

$$u_c[m] = \cos\left(2\pi\frac{f_c}{f_{s,a}}m + \mathcal{F}^{-1}\{L[\Delta m]\}\right). \tag{5.6}$$

In the next step the lowpass signal is multiplied by the carrier to get the bandpass signal:

$$u'_{\mathrm{BP}}[m] = u_{\mathrm{LP}}[m] \cdot u_c[m]. \tag{5.7}$$

There are transient oscillations close to $m = 0$ because of the lowpass filter denoted in Equation (5.2). Thus the signal has to be modified to avoid phase jumps when continued. This takes into account the circular convolution in the measurement path and avoids the Gibbs phenomenon at the beginning and the end of the signal. To incorporate this and using Equation (5.3) gives[1]

$$u_{\mathrm{BP}}[m] = \begin{cases} 0 & m \leqq \frac{D}{2} + \lfloor\frac{3f_{s,a}}{4f_c}\rfloor \\ u'_{\mathrm{BP}}[m] & \frac{D}{2} + \lfloor\frac{3f_{s,a}}{4f_c}\rfloor < m \leqq M \\ 0 & m > M. \end{cases} \tag{5.8}$$

$\frac{f_{s,a}}{f_c}$ are the number of samples of one carrier period. Finally we get

$$u_{\mathrm{BP}}[m] = \begin{cases} 0 & m \leqq \frac{D}{2} + \lfloor\frac{3f_{s,a}}{4f_c}\rfloor \\ (u'_{\mathrm{LP}}[m] \star H_{\mathrm{LP,G}}) \cdot \cos\left(2\pi\frac{f_c}{f_{s,a}}m + \mathcal{F}^{-1}\{L[\Delta m]\}\right) & \frac{D}{2} + \lfloor\frac{3f_{s,a}}{4f_c}\rfloor < m \leqq M \\ 0 & m > M. \end{cases} \tag{5.9}$$

Non-linear system. The modulated bandpass signal $u_{\mathrm{BP}}[m]$ is splitted into two branches in **Figure 5.1a**.. The dashed block contains three subblocks according to Subsection 4.1.1 at Page 52:

1. the non-linear system,
2. a delay of D_0 samples and
3. a highpass filter to remove components close to 0 Hz.

The non-linear system is approximated by

$$u'_{\mathrm{BP}}[m] = \left(c' \cdot u_{\mathrm{BP}}[m] + c'' \cdot u^2_{\mathrm{BP}}[m] + c''' \cdot u^3_{\mathrm{BP}}[m]\right) \tag{5.10}$$

with three coefficients c', c'' and c'''. c'' is proportional to components close to $2f_c$, c''' to $3f_c$. The highpass filter with frequency response H_{HP} delays the signal D_{HP}

[1] The factor $\frac{3}{4}$ is arbitrary.

samples (cf. with the previous paragraph). In the end the output of the non-linear system block

$$u_1[m] = u''_{\mathrm{BP}}[m - D_0 + D_{\mathrm{HP}}/2] + u_{\mathrm{BP}}[m] \tag{5.11}$$

with

$$u''_{\mathrm{BP}}[m] = (u'_{\mathrm{BP}}[m] \star H_{\mathrm{HP}}) \cdot \begin{cases} 1 & D/2 < m < M \\ 0 & \text{else.} \end{cases} \tag{5.12}$$

5.2 Measurement path model

AWGN. Additive white Gaussian noise is added in the first block of the measurement path:

$$u_2[m] = u_1[m] + N_0[m]. \tag{5.13}$$

Take into account that the unit of $N_0[m]$ is normalized like the signal itself.

Gaussian lowpass filter. The shape of the frequency response is Gaussian, therefore the used lowpass filter is Gaussian,

$$u_3[m] = u_2[m] \star H_{\mathrm{LP,G}}, \tag{5.14}$$

with an introduced delay of $D_{\mathrm{LP,G}}$ samples.

Amplifier. The cubic model of an amplifier is according to Equation (4.23) on Page 54

$$u'_4[m] = au_3[m] - cu_3^3[m]. \tag{5.15}$$

a is the linear gain, $c = \frac{4a}{3A_{\mathrm{IIP3}}}$ with $A_{\mathrm{IIP3}} = \sqrt{2R10^{\mathrm{IIP3}/10}/1000}$. R is the reference resistance. The maximum input amplitude $u_{3,\max}$ is the one, where the slope goes to zero:

$$u_{3,\max} = |\sqrt{\frac{a}{3c}}|. \tag{5.16}$$

The correspond maximum output amplitude

$$u_{4,\max} := \frac{2a}{3} u_{3,\max}. \tag{5.17}$$

The input-output relation of the whole amplifier

$$u_4[m] = \begin{cases} u'_4[m] & |u'_3[m]| < u_{3,\max} \\ u_{4,\max} & u'_3[m] \geqq u_{3,\max} \\ -u_{4,\max} & u'_3[m] \leqq -u_{3,\max}. \end{cases} \tag{5.18}$$

With an abuse of notation the pseudo-unit of IIP3 is dBm but all other quantities are normalized, i.e. the units are 1. Therefore we define

$$\frac{\text{IIP3}}{1\ \text{dBm}^*} := 10\log_{10}\frac{1000\frac{A_{\text{IIP3}}^2}{1}}{2R}. \tag{5.19}$$

The input-output-relation of this amplifier model is plotted in **Figure 5.2** with different IIP3 parameters and $R := 1$. The cubic amplifier has been derived using

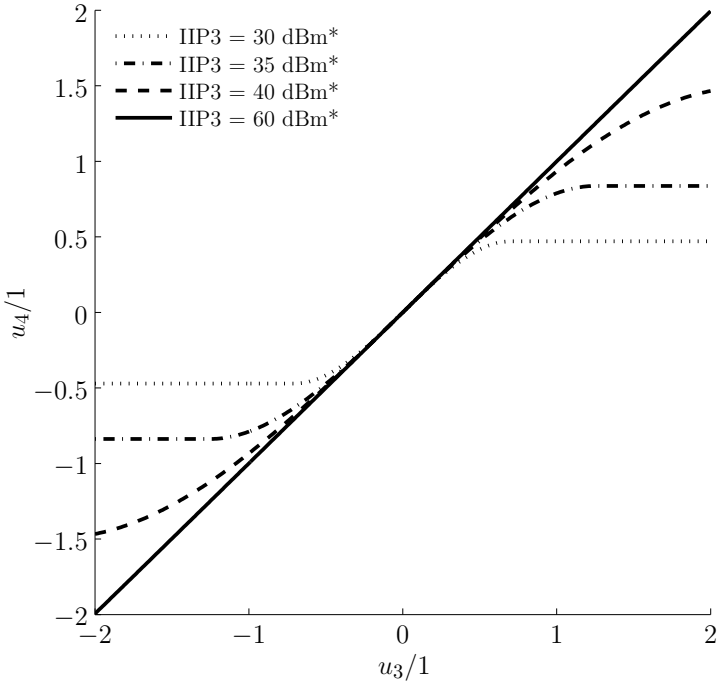


Figure 5.2: Input-output-relation with different values of IIP3. The star (*) denotes that the signal is normalized. The typical interval of the input signal without noise and errors is $[-1; 1]$ in our simulations.

two tones ω_1 and ω_2. An interesting diagram plots the output versus the input power of one fundamental (at ω_1 or ω_2) and one third-order component (at $2\omega_2 - \omega_1$ or $2\omega_1 - \omega_2$) in **Figure 5.3**.

Re-sampler. The implementation of the re-sampler is very basic. The standard deviation of the uniform time jitter is only given in number of samples and a round function discretizes the random time jitter to samples. The ratio or re-sample factor $R \in \mathbb{Q}$ is defined by

$$R := \frac{f_{\text{s,a}}}{f_{\text{s}}} \tag{5.20}$$

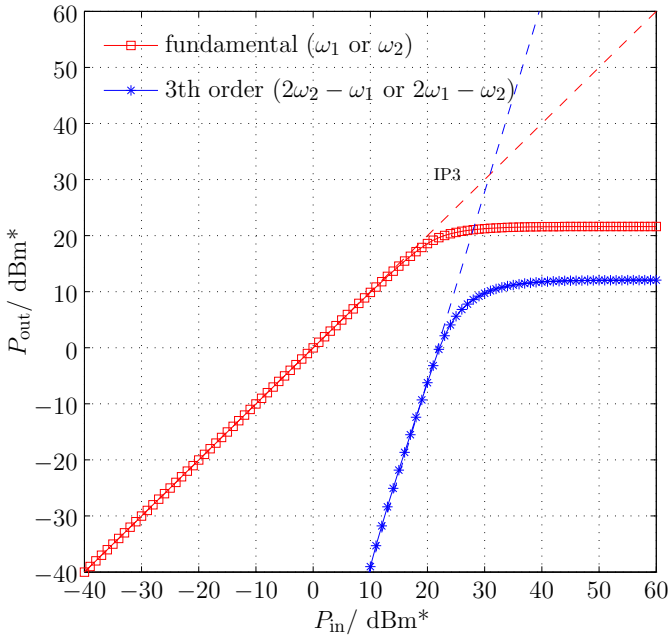


Figure 5.3: Input-power versus output-power simulated with an IIP3 of 30 dBm*.

and the input-output relation:

$$u_5[n] = u_4[nR + [N_{\text{sampler}}]]. \tag{5.21}$$

The inner-brackets denote the round function $[\cdot] : \mathbb{Q}_{64} \to \mathbb{Z}$ and the random variable N_{sampler} has a Binomial distribution which is the approximation of a normal distribution $\mathcal{N}(0, \sigma^2_{\text{jitter}})$ for time-discrete signals.

Quantizer. The Quantizer is described on Page 56 and we recall Equation (4.37) to get

$$u[n] = \begin{cases} -a/2 & u_5[n] \leqq -a/2 \\ [u_5[n]/q]\, q & -a/2 < u_5[n] < a/2 \\ a/2 & u_5[n] \geqq a/2. \end{cases} \tag{5.22}$$

$[\cdot] : \mathbb{Q}_{64} \to \mathbb{Z}$ denotes the round function and $u[n] \in \{-a/2, \cdots, -q = -a/2^{k-1}, 0, q = a/2^{k-1}, \cdots, a/2\} \subset \mathbb{Q}_{64}$.

Hilbert envelope detector. Implementing the filter in the frequency domain has been chosen with the `hilbert` command in Matlab®, thus one gets the input-output-relation from Equation (4.39) on Page 57:

$$r[n] = |\mathcal{F}^{-1}\{\underbrace{\mathcal{F}\{u[n]\}}_{U[n]} \cdot \underbrace{\mathcal{F}\{h[n]\}}_{H[n]}\}| \in \mathbb{Q}_{64}. \tag{5.23}$$

Let N be the length of the signal in samples then — cf. with [hel07] —

$$H[n] = \begin{cases} 0 & n < 0 \\ 1 & n = 0 \text{ and } n = N/2 \\ 2 & 0 < n < N/2. \end{cases} \tag{5.24}$$

Of course this can be enhanced using a different window like an Optimal FIR filter etc. You will see later that this implementation will cause an error floor.

5.3 Reference path

To get an error of the envelope $r[n]$ it has to be compared with an ideal signal. For our purpose this would be $u_{\mathrm{LP}}[m]$. Since the measurement path includes a re-sampler, this has to be done too. It follows a reference path which is plotted in **Figure 5.1b**.

In general the reference signal is undefined, so the source signal is denoted as $u_{\mathrm{ideal}}[m]$. It has to be re-sampled to get the same sampling rate like the detected envelope $r[n]$. Since the measurement path includes delays in the pseudo-analogue part of the path $u_{\mathrm{ideal}}[n]$ has to be delayed too. According to Equation (4.34) the delay has to be multiplied with the re-sample factor R to get the delay D_1 corresponding to the sampling rate f_s:

5.3.1 Implementation

$u_{\mathrm{ideal}}[m]$ is implemented like in the source model, the upper part of Figure 5.1a.

$$u_{\mathrm{ideal}}[m] = \begin{cases} 0 & m \leqq \frac{D}{2} + \lfloor \frac{3 f_{s,a}}{4 f_c} \rfloor \\ [u'_{\mathrm{LP}}[m] \star H_{\mathrm{LP,G}}] & \frac{D}{2} + \lfloor \frac{3 f_{s,a}}{4 f_c} \rfloor < m \leqq M \\ 0 & m > M. \end{cases} \tag{5.25}$$

D is the delay in samples of a Gaussian filter with frequency response $H_{\mathrm{LP,G}}$. M is the length of the signal in samples. The following block re-samples using Equation (5.20):

$$u_{\mathrm{ideal}}[n] = u_{\mathrm{ideal}}[nR]. \tag{5.26}$$

Since the measurement path introduces a delay $D_{\mathrm{LP,G}}$ (cf. with Section 5.2) due to the Gaussian lowpass filter, $u_{\mathrm{ideal}}[n]$ has to be delayed too. The last block considers this:

$$u_{\mathrm{ref}}[n] = u_{\mathrm{ideal}}[n - D_1/2] \tag{5.27}$$

with

$$D_1 = \left[\frac{D_{\mathrm{LP,G}}}{R} \right]. \tag{5.28}$$

$[\,\cdot\,] : \mathbb{Q}_{64} \to \mathbb{Z}$ denotes the round function again.

5.3.2 Error

To measure the error of the whole chain we are using the root-mean-squared error between the outcome of the measurement path ($r[n]$) and the reference path ($u_{\text{ref}}[n]$).

Definition 20 — Root-mean-squared error. Let $\boldsymbol{r} = (\ldots, r[0], r[1], r[2], \ldots)^\dagger$ and $\boldsymbol{u}_{\text{ref}} = (\ldots, u_{\text{ref}}[0], u_{\text{ref}}[1], u_{\text{ref}}[2], \ldots)^\dagger$ be the signal vectors of the detected envelope and the ideal lowpass signal, both with N elements. We define the root-mean-squared error by

$$\varepsilon_{\text{RMS}} := \frac{||\boldsymbol{r} - \boldsymbol{u}_{\text{ref}}||}{\sqrt{N}}. \tag{5.29}$$

An error is often used in decibel which we will consider in this paper:

$$\varepsilon = 20 \log_{10} \left(\frac{||\boldsymbol{r} - \boldsymbol{u}_{\text{ref}}||}{\sqrt{N}} \right) = 10 \log_{10} \sum_{n=1}^{N} \frac{(r[n] - u_{\text{ref}}[n])^2}{N}. \tag{5.30}$$

Note that this isn't the root-mean-squared error anymore, so we are using only ε from now on.

5.3.3 Additional notes to the implementation in Matlab®

All developped equations and relations have to be implemented in Matlab®. We will give some special comments:

- A semi-object-oriented programming style has been chosen to be as compatible as possible. The concept is the same as of an object-oriented programming style but developped with structures and functions.
- This implies that every block can be programmed and tested separately.
- In every simulation three main files exist:
  1. the main file, where all parameters are defined in a structure,
  2. an implementation of the source model,
  3. an implementation of the measurement model and of the reference path.
- Thus different parameters of different simulations need only a change of the source code in the main file.
- A verbose mode makes it possible to few the time signal and spectrum of most signals (simulation steps) in the paths.

A coarse flowchart is shown in **Figure 5.4**.

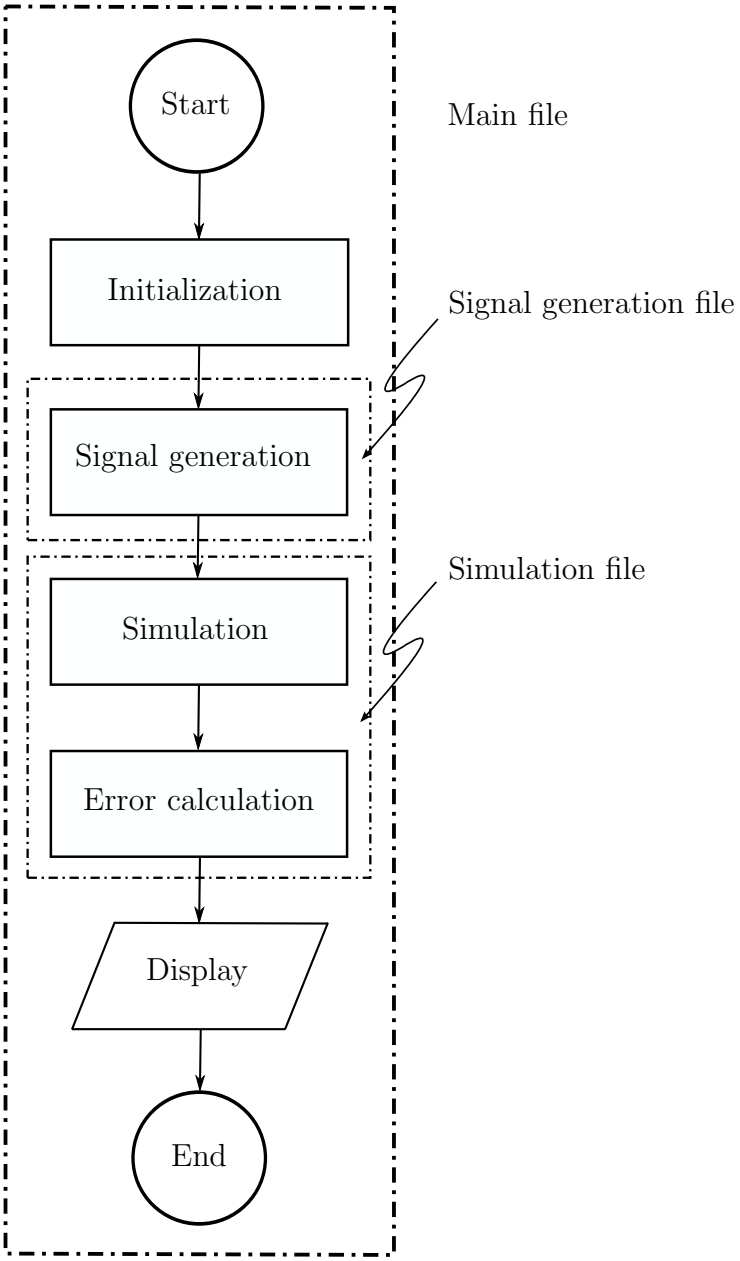


Figure 5.4: Coarse flowchart of the developed simulations. Only the "Initialization" and "Display" parts are being changed for each different simulation. "Signal generation" consists of several blocks for generating a defined bandpass signal, adding phase noise and adding harmonics with the non-linear system. "Simulation" contains the measurement paths, where every block of Blockdiagram 5.1a corresponds to one block in this flowchart.

6 Common setup of source model

The parameters for all following simulations are printed in **Table 6.1** for the signal source model. If a simulation uses another parameter it will be listed separately.

parameters of source model	value
type	type A
carrier frequency f_c	13.56 MHz
pseudo sampling rate[a] $f_{s,a}$	13.56 GS/s
simulated sequences	10 times "Sequence X"
data rate	$f_c/16 = 847.5$ kbit/s
t_b	$16/f_c$
t_x	$8/f_c$
t_1	$5/f_c$
t_5	$4/f_c$
t_6	$6/f_c$
a	0.6
carrier[b]	0 dBc $:\leftrightarrow c' = -300$ dB
2nd harmonic	-40 dBc $\leftrightarrow c'' = 10$ dB
3rd harmonic	-50 dBc $\leftrightarrow c''' = -10$ dB
D_0	0
Shape[c] of $L(\Delta\omega)$	(1 kHz, -113.92 dB), (10 kHz, -122.72 dB), (100 kHz, -130.87 dB), (1 MHz, -140.23 dB)

[a] GS/s specifies 10^9 samples per second.
[b] c'' is the second harmonic, c''' the third. dBc denotes decibels relative to carrier.
[c] Single sideband noise spectral density; shape specified via tuples: (offset from f_c, amplitude)

Table 6.1: Parameters of the source model.

6.1 Spectrum of the signal produced by the source model.

The spectrum of $u_1[m]$ with parameters specified in Table 6.1 is plotted in **Figure 6.1** between 0 Hz and 50 MHz. The signal components close to 27.12 MHz and

40.68 MHz are the effect of the non-linear system in the source model.

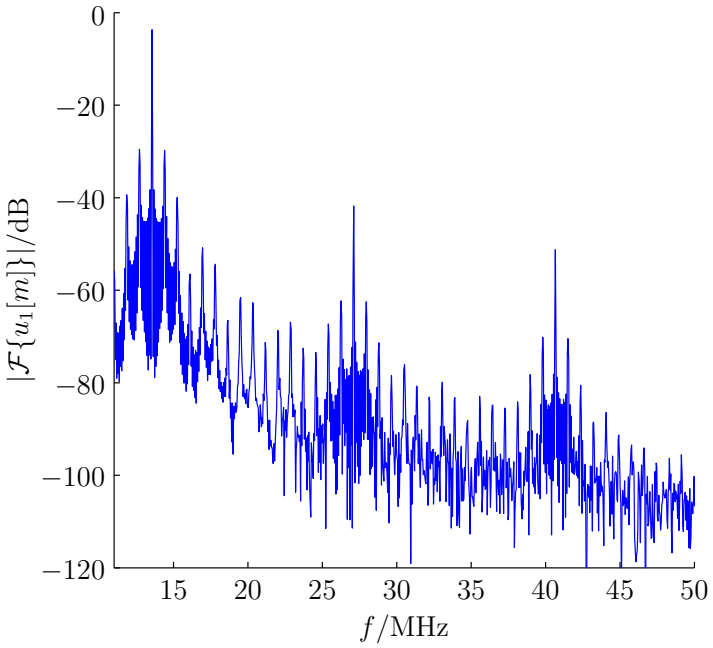


Figure 6.1: Spectrum of modulated source with amplitude of harmonics of −40 dBc and −50 dBc.

6.2 Example: Agilent's DSO7052A

As an example we are using an oscilloscope of Agilent (DSO7052A with passive probe 10073C, specified in [Agi09]) and we choose the parameters of the measurement path like in **Table 6.2**. The bandwidth of the Gaussian lowpass filter has been chosen using Equation (3.10) on Page 47 which takes into account the used probe. For the sake of simulation the pseudo sampling rate is set to 80 GS/s. Since the noise of the oscilloscope is 0.5% of "full screen" we assume that the signal fills the display. Therefore the noise is chosen to be 0.5% of the normalized signal. The same assumption is made for the limit of the amplitude of the quantizer.

The result is plotted in **Figure 6.2** without 2^{nd} and 3^{th} harmonics, a zoomed version with 2^{nd} and 3^{th} harmonics in **Figure 6.3** and the error of the latter one separately in **Figure 6.4**. The value of the whole error is (cf. with Equation (5.30))

$$\varepsilon \approx \begin{cases} -51\ \mathrm{dB} & \text{without harmonics} \\ -41\ \mathrm{dB} & \text{with } 2^{\text{nd}} \text{ and } 3^{\text{th}} \text{ harmonics } (-40\ \mathrm{dBc} \text{ and } -50\ \mathrm{dBc}). \end{cases} \tag{6.1}$$

setup of measurement path	value
Gaussian noise:	
μ	0
σ^2	$(0.005)^2$
$f_{-3\,\mathrm{dB}}$ of Gaussian LP filter	353.55 MHz
non-linearities of oscilloscope:	
IIP3	100 dBm (arbitrary)
reference resistance	1
sampling rate (re-sample)	4 GHz
quantizer:	
$\sigma^2_{\mathrm{jitter}}$	$1.2 \leftrightarrow 15$ ps
quantization bits	8 bit
limit of amplitude	1.1

Table 6.2: Setup of the simulations of Agilent's DSO7052A [Agi09].

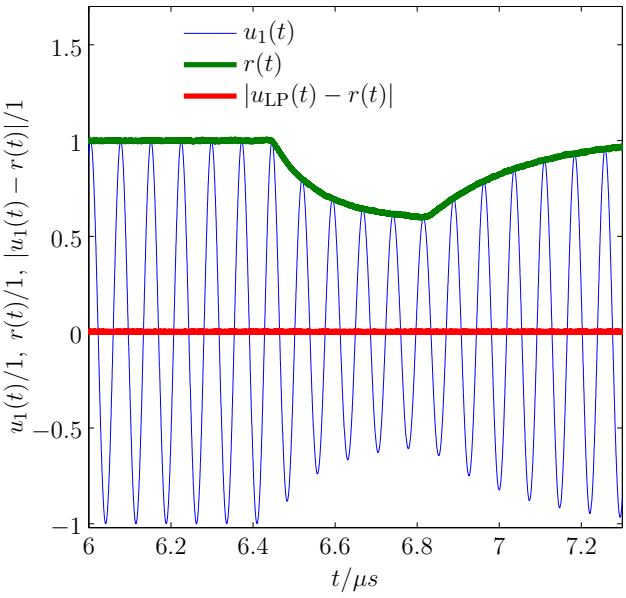


Figure 6.2: Input signal of measurement path ($u_1(t)$) without second and third harmonic, detected envelope $r(t)$ and resulting error $|u_1(t) - r(t)|$.

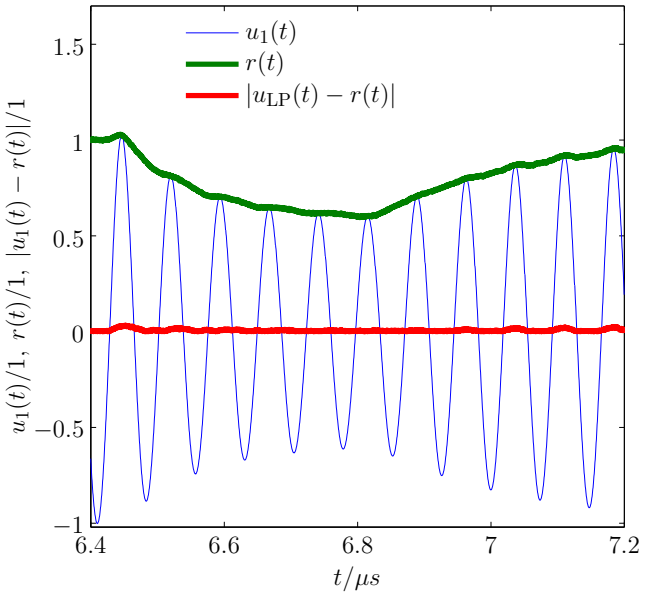


Figure 6.3: Input signal of measurement path ($u_1(t)$) with 2nd and 3th harmonics, detected envelope $r(t)$ and resulting error $|u_1(t) - r(t)|$. The error is very small and is mainly due to harmonics close to 27.12 MHz and the Gibbs phenomenon.

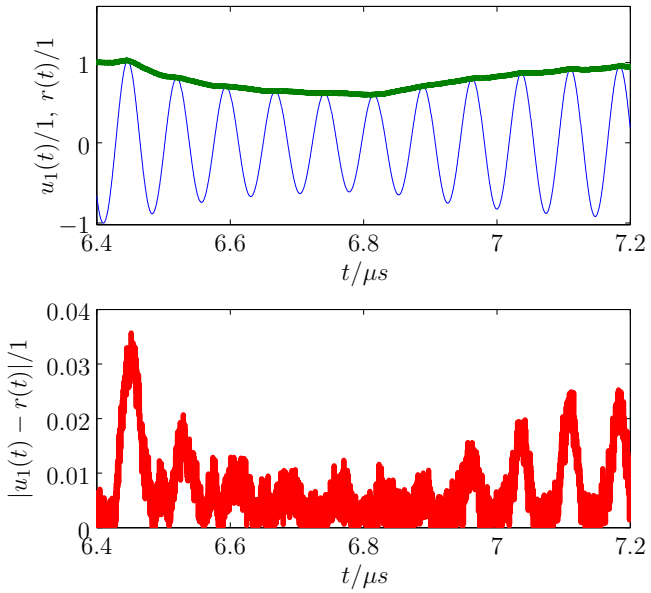


Figure 6.4: Error of measurement path ($u_1(t)$) with 2nd and 3th harmonics, detected envelope $r(t)$ and resulting error $|u_1(t) - r(t)|$. The error is very small and is mainly due to harmonics close to 27.12 MHz and the Gibbs phenomenon.

7 Results of simulation and Conclusions

Let us assume an ideal system with no noise, no time jitter, no quantization noise etc. What will happen if one parameter is switched on, like the noise? Or if the sampling rate changes? This and some other questions will be asked in this chapter, and with corresponding simulations and error curves will be presented.

7.1 Influence of harmonics to the received signal

The influence of the first and second harmonic to the error is plotted at **Figure 7.1**. The simulation sweeps one harmonic while the other one is set to $-\infty$ dBc. The diagram shows an error floor at -60 dB mainly caused by the Gibbs phenomenon of the Hilbert envelope detector.

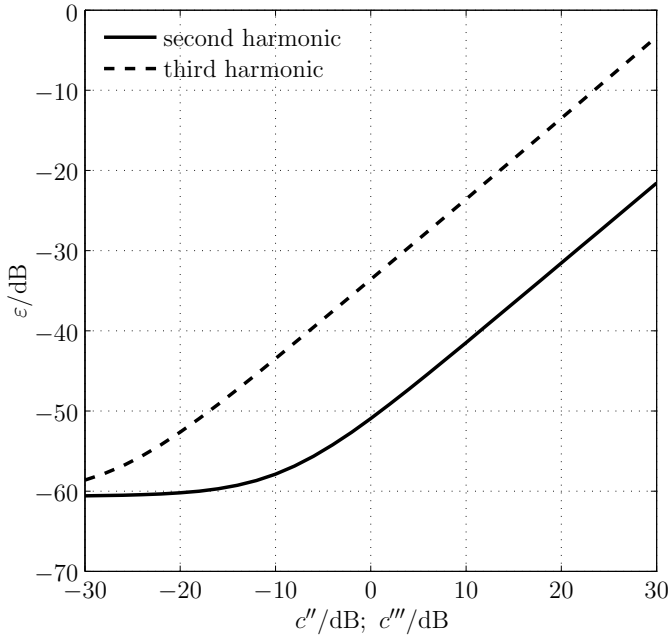


Figure 7.1: Error of different second and third harmonics.

7.2 Influence of the intermodulation within the oscilloscope

The non-linearities in the oscilloscope cause intermodulation. We have introduced the cubic model which has the IIP3 as the parameter of non-linearity (cf. Equation (5.15) on Page 61). The diagram in **Figure 7.2** shows the error vs. IIP3. Note that in the simulation a normalized signal is used, hence the unit of the amplitudes is 1. The same applies for the reference resistance R. In the following $c'' = 10$ dB doesn't only imply a 2^{nd} harmonic but a 3^{rd} as well. $c'' = -300$ dB results in source model with switched off non-linearities.

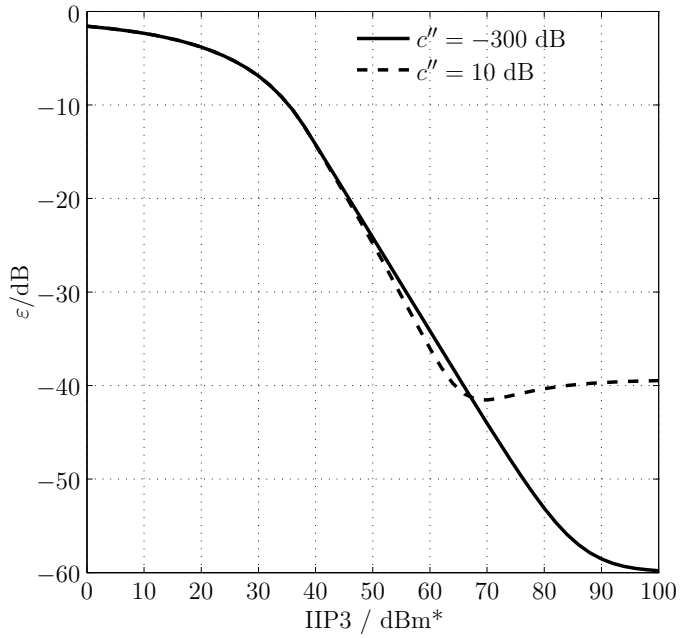


Figure 7.2: The error vs. IIP3.

7.3 Influence of the sampler

The influence of the (re-)sampler is plotted in **Figure 7.3** with $f_{s,a} = 54.24$ GS/s. The curves are zigzagging due to the random sample phase. Anyway, it gives a good impression where the mean of the errors is close to the error floors.

Later we will discuss different bandwidths of the Gaussian filter using different sampling rates and will see that even if the differences aren't great, higher data rates will improve the result.

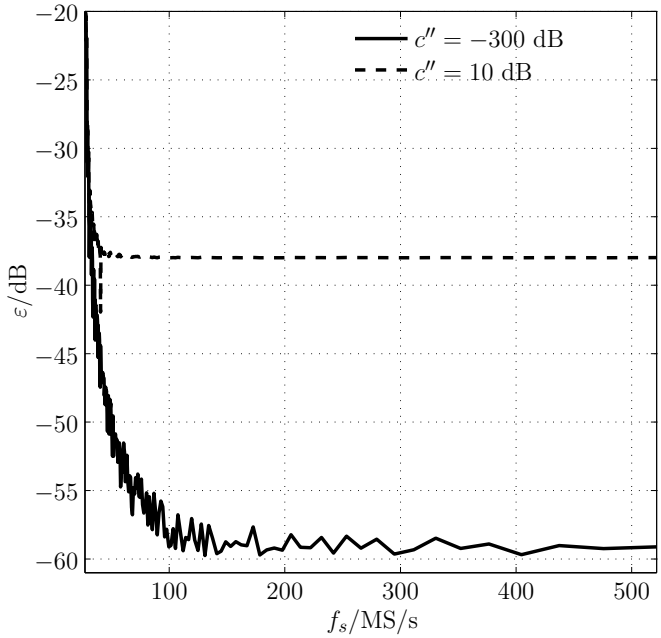


Figure 7.3: Error introduced by different sampling rates.

7.4 Influence of Gaussian noise

The error of different Gaussian noise is plotted in **Figure 7.4**. The assumption is that the mean is zero ($\mu = 0$). Since the signal-to-noise ratio (SNR) is sometimes more interesting than the used mean and variance of the noise, one has to calculate it by

$$\text{SNR} := \frac{P_{\text{signal}}}{P_{\text{noise}}} = \frac{\sigma_X^2 + \mu_X^2}{\mu^2 + \sigma^2} = \frac{\sigma_X^2 + \mu_X^2}{\sigma^2}. \tag{7.1}$$

σ_X^2 and μ_X are the variance and the mean of the signal.

With SNR = 40 dB it follows a mean squared error of about −40 dB.

7.5 Influence of quantization

What should be the lower bound of the number of bits of the quantizer? **Figure 7.5** shows the introduced error for different numbers of bits and different variances σ^2 of Gaussian noise $N_{\text{o}}[m] \sim \mathcal{N}(0, \sigma^2)$. The SNR is calculated using Equation (7.1). Note that an SNR $\approx$ 50 dB is a typical value for an oscilloscope. Therefore a quantizer of 8 bit works quite well in most cases.

A quantizer should have at least 8 bit since the mean squared error is about −50 dB without noise and harmonics.

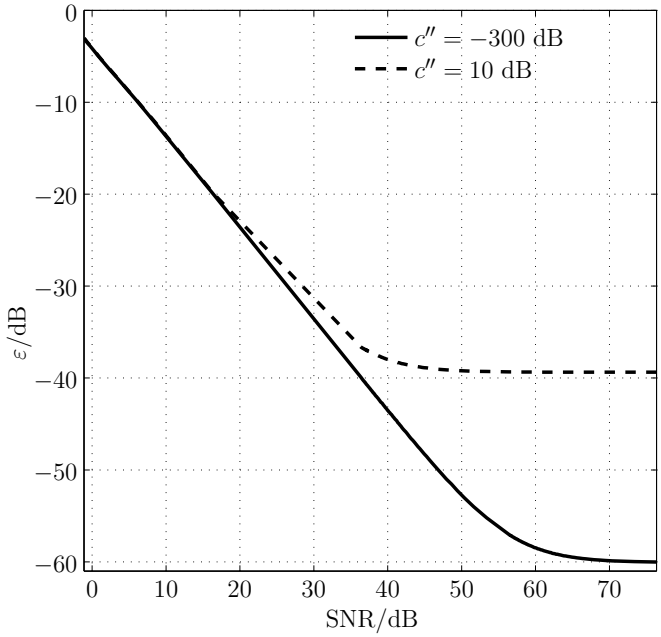


Figure 7.4: Error introduced by Gaussian noise.

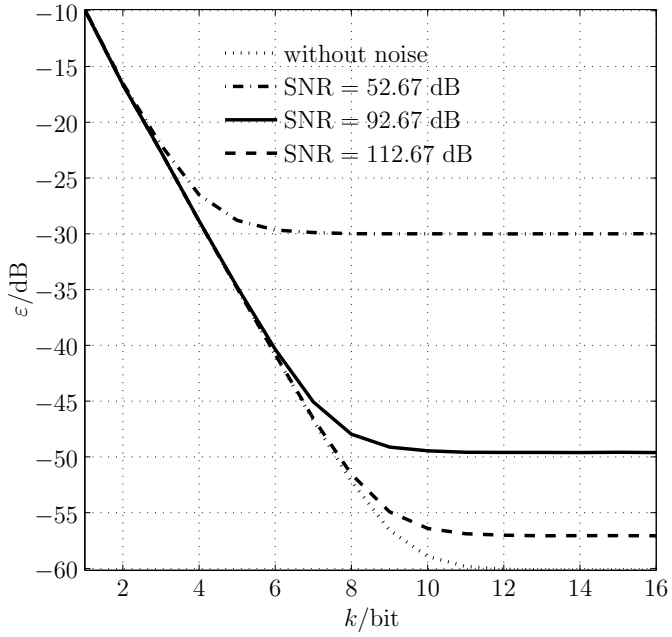


Figure 7.5: Quantization error for different Gaussian noise.

7.6 Influence of the time jitter of the sampler

We assume a Gaussian distribution of the time jitter. The simulation is done with a time-discrete signal, hence this distribution has to be discretized. It is called a Binomial distribution. **Figure 7.6** shows the introduced error which increases very fast with increasing time jitter. If the trigger of the sampler has a time jitter of about 15 ps the error will be about −50 dB.

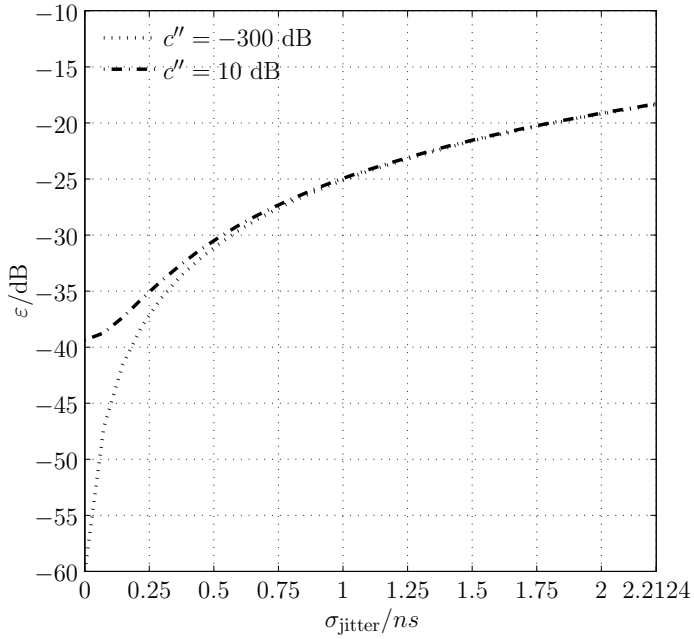


Figure 7.6: Error introduced by time jitter.

7.7 Influence of delay of harmonics

In the previous sections we have assumed that the signals of second and third harmonics aren't delayed. But how does a delay influence the error? **Figure 7.7** gives the answer.

7.8 Influence of the frequency response of the oscilloscope

The lowpass filter has a Gaussian shape in the frequency response; thus the influence of the bandwidth is interesting. **Figures 7.8 to 7.11** show different simulations depending on noise (SNR ≈ -53 dB is a typical value for an oscilloscope) and

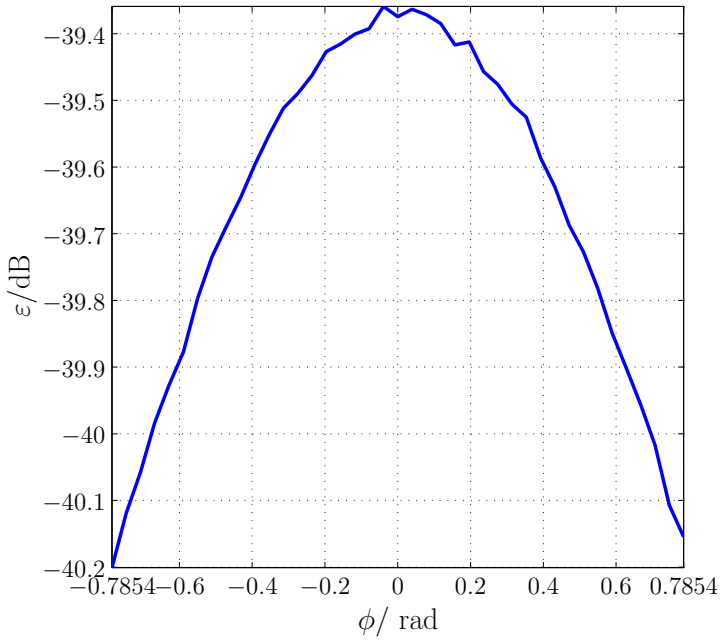


Figure 7.7: Different phase shifts of second and third harmonics.

harmonics. For these simulations the pseudo sampling rate has been set to $f_{s,a} = 54.24$ GS/s.

Illustration of the figures.

1. In Figure 7.8 the influence of the bandwith is shown without harmonics and noise. Note the error at lower bandwidths owing to asymmetrical[1] attenuation of the signal.

2. The following Figure 7.9 plots the same simulation with noise. As a result the error increases at higher bandwidths again.

3. Figure 7.10 shows the simulation without noise but with second and third harmonics. Again, we see an increasing error at higher frequencies due to the harmonics. Thus the error tends to a constant (= new error floor).

4. Both, harmonics and noise lead us to Figure 7.11.

Enhanced simulations. By comparing the figures with second and third harmonics as a rule of thumb one can specify $f_{-3\,\mathrm{dB}} > 50$ MHz. In **Figures 7.12 and 7.13**

[1] Asymmetrical means that the signal components with a frequency $f < f_c$ are attenuated less than the signal components with a frequency $f > f_c$.

the simulation is done for four different sampling rates (the sampler uses random sample phases). It is obvious that higher sampling rates improve the result. As a rule of thumb for measurements with second and third harmonics — according to Equation (3.12) the sampling rate should beat least four times the bandwidth of the Gaussian filter —, the sampling rate should be > 200 MS/s:

$$\max\left(\varepsilon(200\ \text{MS/s}) - \varepsilon(400\ \text{MS/s})\right) < 1.52\ \text{dB}. \tag{7.2}$$

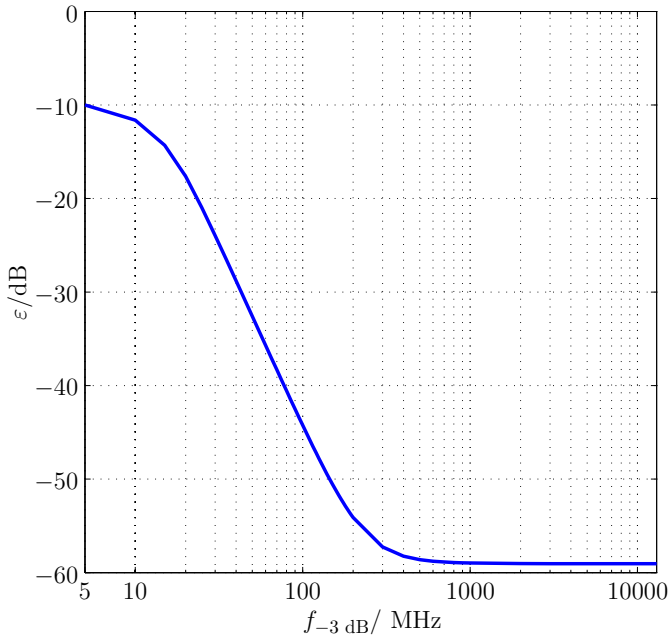


Figure 7.8: Influence of different bandwidths of Gaussian lowpass filter without second and third harmonics and without noise.

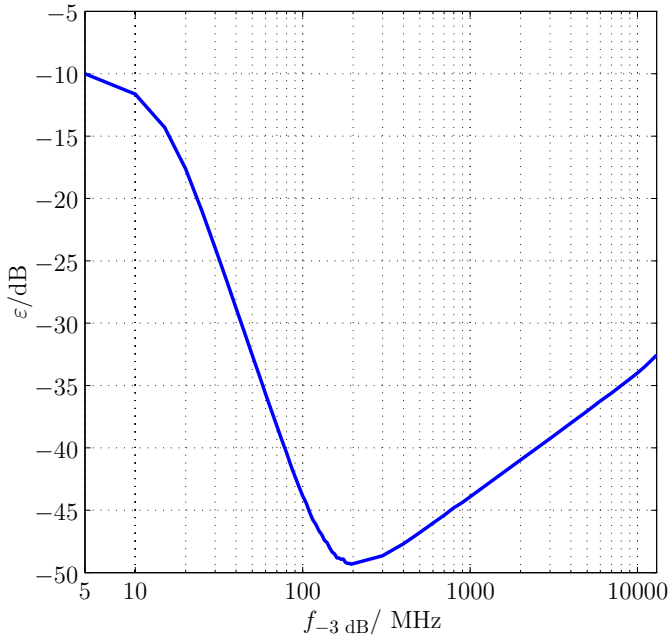


Figure 7.9: Influence of different bandwidths of Gaussian lowpass filter without second and third harmonics and with noise.

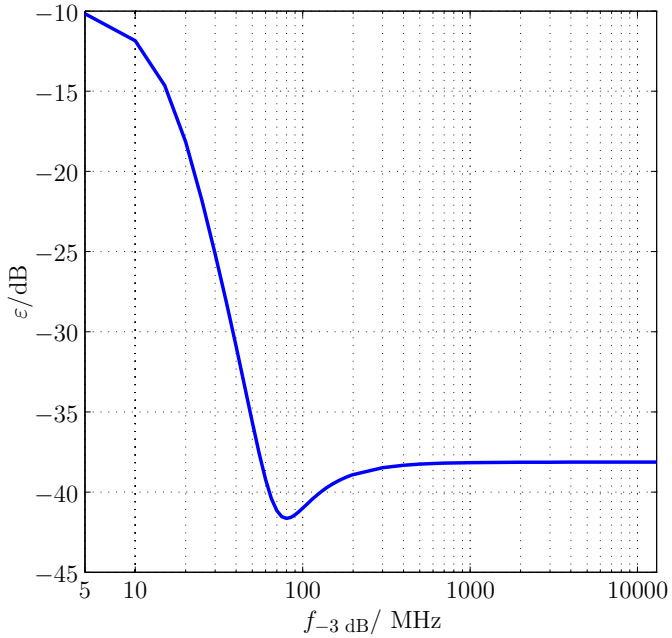


Figure 7.10: Influence of different bandwidths of Gaussian lowpass filter with second and third harmonics and without noise.

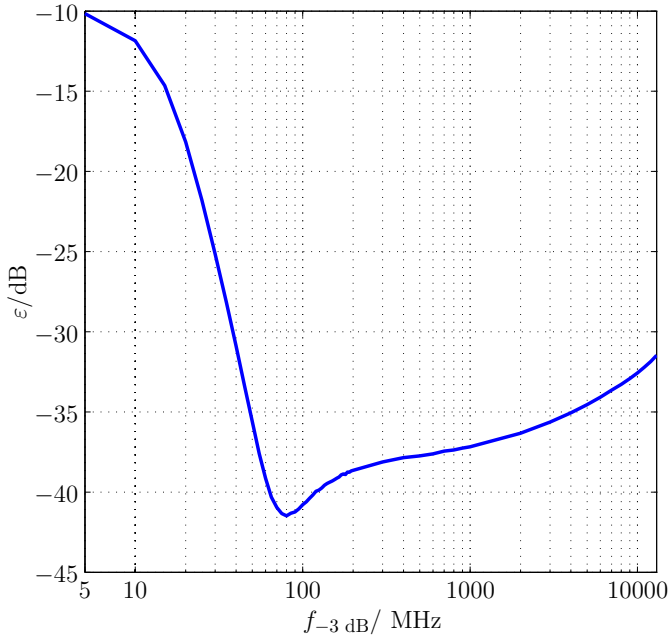


Figure 7.11: Influence of different bandwidths of Gaussian lowpass filter with second and third harmonics and with noise.

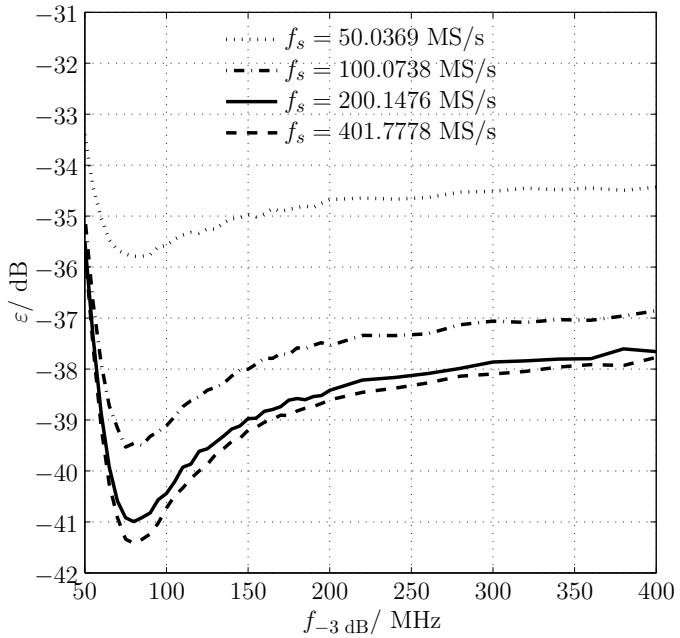


Figure 7.12: Influence of different bandwidths of Gaussian lowpass filter with second and third harmonics, noise and four different sample frequencies. Note that the sample phase is random.

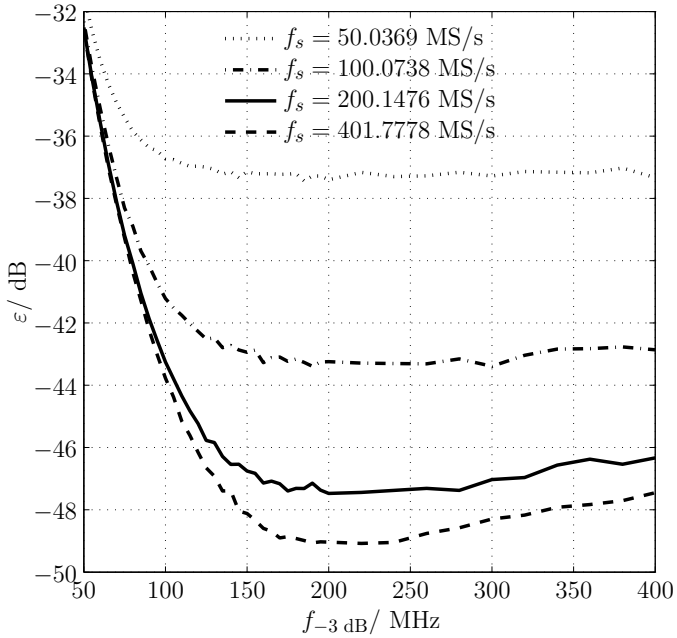


Figure 7.13: Influence of different bandwidths of Gaussian lowpass filter with noise and four different sample frequencies. Note that the sample phase is random.

8 Summary and conclusion

The postulated source code (cf. with Appendix D) of the envelope detector based on Hilbert transform can be improved very easily by

- using another FFT implementation like FFTW,
- reducing the Gibb's phenomenon and
- enhancing the allowed length of the signal.

With an implementation in the time domain, using, for example, an Optimal FIR filter the latter two items could be easily managed. Note that the sample period has to be very accurate because the bandpass signal is sampled directly.

Comparing all figures from the previous chapter shows an important fact. There exists an error floor mainly due to Gibbs phenomenon depending on the harmonics:

harmonics	error floor
first	−60 dB
first, second and third	−40 dB

In addition following facts are:

- The non-linearities of the measurement path have to be very small.
- From the noise of a typical oscilloscope the lower bound of quantization bits follows with 8 bit.
- The measurement result is very sensitive to time jitter.
- As a rule of thumb $f_{-3\,\mathrm{dB}} > 50\,\mathrm{MHz}$ if the frequency response of the measurement path has a Gaussian shape. Note that there is a minimum of error close to 75 MHz.
- Another rule of thumb exists for the sampling frequency, which should be $> 200\,\mathrm{MS/s}$.

A Confidence Interval

The value of the confidence interval of a random variable X specifies the probability $P(a \leqq X \leqq b)$ that a realization x of the random process falls into the interval $[a; b]$. It can be calculated if the pdf $f_X(x)$ of the random value X is known with

$$\int_a^b f_X(x)dx = P(a \leqq X \leqq b).$$

If the probability density function has a Gaussian distribution the interval is defined symmetrically around the mean. Since the pdf of a Gaussian random value is only defined by the first and second order properties, the mean and variance are used to define the confidence interval (CI):

Probability of values in CI	Interval
95%	$\mu \pm 1.28155 \cdot \sigma$
99.99999980%	$\mu \pm 6 \cdot \sigma$

Note that in our case the CI is estimated using L different realizations. Thus μ and σ must be replaced with $\hat{\mu}$ and $\hat{\sigma}$. The *law of large numbers* has to be considered which tells that if $L \to \infty$ the estimation will tend close to the real-world stochastic properties.

B Pulse amplitude modulation

Recall that the signal which is sent over a channel in the baseband system is defined as

$$u_{LP}(t) = \sum_{k=-\infty}^{\infty} a[k] g_k(t - kT)$$

where $a[k] \in \{s^{(n)}\}$ is the complex-valued symbol at time index k and $g_k(t)$ is the impulse response of the time continuous *transmit filter* with length of T. In our examples the pulse shapes are rectangular and are defined in [ISO08] (Sequences Z, X, Y). Note that rectangular pulse shapes aren't possible in the real world since the transmit bandwidth has to be infinite. For real-valued bandpass signals one needs a lowband $\leftrightarrow$ bandpass transform

$$u(t) = \mathrm{Re}\left(\sqrt{2} u_{LP}(t) e^{j\omega_0 t}\right) = \frac{\sqrt{2}}{2}(u_{LP}(t) e^{j\omega_0 t} + u_{LP}^*(t) e^{-j\omega_0 t})$$

where $f_c = \omega_c / 2\pi$ is the carrier frequency.

C Rise time, settle time and overshoot

A jump function with oscillations close to the point of discontinuity like in **Figure 2.14** on Page 36 can be written as a function

$$f(t) := f_\infty(1 + e^{j\omega t}e^{-\rho t})\sigma(t) \tag{C.1}$$

with the unit jump

$$\sigma(t) := \begin{cases} 1 & t \geqq 0 \\ 0 & \text{elsewhere} \end{cases}, \; f_\infty := \lim_{t\to\infty} f(t) \tag{C.2}$$

and an attenuation factor $\rho < 1$. f_∞ is called the final value of $f(t)$. Then the settle time

$$t_{s,2\%} := t|_{|f(t)-f_\infty|=0.02f_\infty}. \tag{C.3}$$

It means, that the difference between the oscillation and the desired final value f_∞ is 2% of the desired value. This property is important, to know how fast the oscillation reduces. The ratio

$$o := \max f(t)/f_\infty \tag{C.4}$$

is called the overshoot of a step jump. Removing is impossible. The rise time is the time $f(t)$ takes from 10% to 90% of the final value f_∞ of the function,

$$t_r := f^{-1}(0.9f_\infty) - f^{-1}(0.1f_\infty), \tag{C.5}$$

where $f^{-1}(\,\cdot\,)$ is the inverse of $f(\,\cdot\,)$. Note: Since in our case a discrete-time signal is analyzed, both time properties are measured in samples. In our case it is multiplied by the sample period to compare it with the continuous-time signal, thus it is an approximation.

D Source Code

For the sake of completeness the source code of the envelope detector in Annex 7 of [ISO] is re-printed.

D.1 (I)FFT

D.1.1 Header

```
#ifndef FFTRM_H
#define FFTRM_H
#define RE(z) ((z).r)
#define IM(z) ((z).i)
typedef float real;
typedef double doublereal;
typedef struct { real r, i; } complex;
typedef struct { doublereal r, i; } doublecomplex;
int zffts (int debug, doublecomplex *X, int M);
int ziffts (int debug, doublecomplex *X, int M);
void zfftrmc(doublecomplex *X, int M, int P, float D);
void rmpo (int *rv, int *rvp );
#endif
```

D.1.2 Source file

```
#include <stdio.h>
#include <math.h>
#include <malloc.h>
#include "fftrm.h"
/* #ifndef M_PI */
#define M_PI 3.14159265358979323846264338327 95
/* #endif */
float *WR;
float *WI;
doublereal *DWR;
doublereal *DWI;
void rmpo( int *rv, int *rvp )
{
 int value_h;
 int n;
 n    = 1;
```

```
17  *rvp = -1;
    value_h = 1;
19  while ( value_h > 0 ) {
    value_h = *rv - n;
21  (*rvp)++;
    n += n;
23  }
}
25 void zfftrmc( doublecomplex *X, int M, int P, float D )
{
27  int MV2,MM1,J,I,K,L,LE,LE1,IP,IQ,IND,IND1,R;
    int I1,J1,IPOTR;
29  float A,B;
    float WCOS,WSIN;
31  float VR,VI;
    float ARG;
33  static int IPOTC;
    static float DALT;
35  IPOTR = 0;
    DWR = (doublereal *)calloc(M,sizeof(doublereal));
37  DWI = (doublereal *)calloc(M,sizeof(doublereal));
   /* if (IPOTC == P & D == DALT) goto warmstart; */
39  IPOTC = P;
    DALT     = (float)D;
41  LE = 1;
    IND = 0;
43  for (L=1;L<=P;L++) {
    LE1 = LE;
45  LE = LE*2;
    DWR[IND] = 1.0;
47  DWI[IND] = 0.0;
    ARG= (float)M_PI/(float)LE1;
49  WCOS = (float)cos(ARG);
    WSIN = (float)(D*sin(ARG));
51  for (R=1;R<=LE1;R++) {
    IND1 = IND+1;
53  A = (float)DWR[IND];
    B = (float)DWI[IND];
55  DWR[IND1] = A*WCOS - B*WSIN;
    DWI[IND1] = B*WCOS + A*WSIN;
57  ++IND;
    }
59  }
    /* warmstart: */
61  MV2=M/2;
    MM1=M-1;
63  J=1;
    for (I=1; I<=MM1; I++) {
65  if (I >= J)
    goto P1;
67  J1   = J-1;
```

```
    I1    = I-1;
69  VR    = (float)RE(X[J1]);
    VI    = (float)IM(X[J1]);
71  RE(X[J1]) = RE(X[I1]);
    IM(X[J1]) = IM(X[I1]);
73  RE(X[I1]) = VR;
    IM(X[I1]) = VI;
75 P1: K      = MV2;
   P2: if (K >= J) goto P3;
77  J = J-K;
    K = K/2;
79  goto P2;
   P3: J = J+K;
81  }
    IND = 0;
83  LE = 1;
    for (L=1; L<=P; L++) {
85  LE1 = LE;
    LE = LE*2;
87  for (R=0; R<LE1; R++) {
    WCOS = (float)DWR[IND];
89  WSIN = (float)DWI[IND];
    IND = IND+1;
91  for (IQ=R; IQ<M; IQ+=LE) {
        IP = IQ+LE1;
93      A = (float)RE(X[IP]);
        B = (float)IM(X[IP]);
95      VR = A*WCOS - B*WSIN;
        VI = B*WCOS + A*WSIN;
97      RE(X[IP]) = RE(X[IQ]) - VR;
        IM(X[IP]) = IM(X[IQ]) - VI;
99      RE(X[IQ]) = RE(X[IQ]) + VR;
        IM(X[IQ]) = IM(X[IQ]) + VI;
101 }
    }
103 }
    free(DWR);
105 free(DWI);
   }
107
   int zffts( int debug, doublecomplex *X, int M )
109{
    int P;
111 float D;
    D = -1.0;
113 rmpo( &M, &P);
    if ( debug ) {
115 printf("P = %d\n",P);
    printf("FFT ...\n");
117 }
    zfftrmc( X, M, P, D); /* fftrm.c */
```

```
119  return 0;
     }
121
     int ziffts( int debug, doublecomplex *X, int M )
123  {
      int i;
125   int P;
      float D;
127   D = 1.0;
      rmpo( &M, &P);
129   if ( debug ) {
      printf("P = %d\n",P);
131   printf("IFFT ...\n");
      }
133   zfftrmc( X, M, P, D); /* fftrm.c */
      /*___Multiply with 1/M____*/
135   for (i=0; i<M; i++) {
      RE(X[i]) /= (doublereal)M;
137   IM(X[i]) /= (doublereal)M;
      }
139   return 0;
     }/*End of fftrm.c*/
```

D.2 Hilbert detector

The following listing is using the FFT from above and calculates the envelope of the modulated carrier. The comments haven't been included in the listings:

- It need an input file which has to contain two columns to describe the modulated signal. The first describes the time, the second the correspond amplitude of the signal. The time difference between the time points has to be equidistant.
- The compiled program has to be started at the command line by the program name followed by the input file (standard name of input file is input.txt.). The result is written to output.txt.

```
 1 # include <stdio.h>
   # include <math.h>
 3 # include <malloc.h>
   #include <ctype.h>
 5 #include <string.h>
   # include "fftrm.h"
 7 #define MAX_POINT 5000
   #define M_PI 3.14159265358979323846264338327 95
 9 int debug=0;
   int fftdebug=0;
11 double *Gvalue;
```

```
double *Gtime;
double *Gr;
double *Gi;
double **G; /*Phase Changed*/
double *Gc;
doublecomplex *Gt_ifft;
/*File containing the input data*/
char *InputFileName ="input.txt" ;
/*This function reads the sampled data recorded in the file*/
int ReadData(void);
/*This function function performs the fourier transform*/
void Fft(void);
/*This function performs the necessary phase shift*/
void PhaseShifting(void);
/*This function performs the inverse fourier transfor*/
void Ifft(void);
/*Envelope reconstruction is done by this function*/
int EnvelopeReconstruction(void);
/*For fourier and inverse fourier transformation these two functions
   are used */
/*These functions are defined in fftrm.c */
int zffts ( int debug,doublecomplex *X,int M ); /*Defined in fftrm.c*/
int ziffts( int debug,doublecomplex *X,int M ); /*Defined in fftrm.c*/
int SampledPoints=0;
int N;
int row;
const int col=2;
int ReadData(void)
{
    float a,b;
    int i=0,num1;
    FILE *fp1;
    i=0;
    if ((fp1 = fopen(InputFileName,"r")) == NULL)
    {
        printf("Cannot open input file.\n");
        return 1;
    }
    printf("\nReading data from file ... ... ...%s\n",fp1);
    while(!feof(fp1))
    {
        fscanf(fp1,"%e,%e\n", &a, &b);
        Gtime[SampledPoints] = a;
        Gvalue[SampledPoints] = b;
        SampledPoints++;
        if (SampledPoints>= MAX_POINT) break;
    }
    fclose(fp1);
    fp1=fopen("inputfile.txt","w");
 if (!fp1)
    {
```

```
 fprintf(stdout,"Cann't write the sampled data in inputfile.txt. \n");
63          return 1;
        }
65  for(i=0; i<SampledPoints; i++)
    fprintf(fp1,"%e\n",Gvalue[i]); /*Gtime[i] has been omitted*/
67  fclose(fp1);
        if(debug)
69      {
        fp1=fopen("inputtime.txt","w");
71  if (!fp1)
        {
73  fprintf(stdout,"Cann't write the sampled data in inputtime.txt. \n");
            return 1;
75      }
    for(i=0; i<SampledPoints; i++)
77  fprintf(fp1,"%e\n",Gtime[i]); /*Gtime[i] has been omitted*/
    fclose(fp1);
79      }
        if(debug)
81      {
        if((fp1=fopen("inputfile.bin","wb"))!=NULL)    {
83          num1=fwrite(Gvalue,sizeof(double),SampledPoints,fp1);
        fclose(fp1);
85      }
        }
87      if(SampledPoints<N)
        {
89          for(i=SampledPoints;i<=N;i++)
            {
91              Gvalue[i] = 0;
            }
93      }
        fprintf(stdout,"\nInput file name = %s\n",InputFileName);
95  fprintf(stdout,"Number of sampled data = %d\n",SampledPoints);
        return 0;
97  }/*End Of Function ReadData;*/
    void Fft(void)
99  {
        doublecomplex *Gt_freq;
101     FILE *fp1,*fp2,*fp3;
        int k,num1,num2,num3,z1;
103
        Gt_freq = (doublecomplex *)calloc(sizeof(doublecomplex),row);
105     printf("\nPerforming FFT ... ... ...\n");
        /* FFT Procedure Starts for Sampled Data*/
107     for(k=0;k<=N;k++){
            RE(Gt_freq[k])=Gvalue[k];
109         IM(Gt_freq[k])=0.0;[150,100,90, 70,50,6,5]
        }
111     if(debug){
            if((fp3=fopen("f.bin","wb"))!=NULL) {
```

```
113             num3=fwrite(Gvalue,sizeof(double),row,fp3);
                fclose(fp3);
115             }
        }
117     z1=zffts(fftdebug,Gt_freq,row);/*FFT is done in spatial coordicate
            */
        for (k=0;k<=N;k++) {
119             Gr[k]=RE(Gt_freq[k]);
                Gi[k]=IM(Gt_freq[k]);
121     }
    /* FFT Procedure Ends for Sampled Data*/
123 /* Writing The Real And Imaginary Part Of Reflected Part for Debuging
        */
    /* Writing the real part of sampled data*/
125     if(debug) {
            if((fp1=fopen("Gr.bin","wb"))!=NULL){
127                 num1=fwrite(Gr,sizeof(double),row,fp1);
                    fclose(fp1);
129             }
                else
131                 fprintf(stdout,"Cann't Open Gr.bin");
            /* Writing the img part of sampled data */
133         if((fp2=fopen("Gi.bin","wb"))!=NULL) {
                    num2=fwrite(Gi,sizeof(double),row,fp2);
135                 fclose(fp2);
            }
137         else
                    fprintf(stdout,"Cann't Open Gi.bin");
139         fprintf(stdout,"Num of Real Part Data after FFT = %d\n",num1);
                    fprintf(stdout,"Num of Img Part Data after FFT = %d\n",
                        num2);
141     }
    free(Gt_freq);
143 }/* End Of The Function Fft */
    void PhaseShifting(void)
145 {
        double *tempr, *tempi;
147     int k,num1;
        FILE *fp1;
149     printf("\nPerforming phase shift ... ... ...\n");
        tempr = (double *)calloc(sizeof(double),row);
151     tempi = (double *)calloc(sizeof(double),row);
        for ( k=0; k<=N; k++ )
153     {
            tempr[k]=Gr[k];
155         tempi[k]=Gi[k];
        }
157     for ( k=0; k<=ceil(N/2); k++ )
        {
159         Gr[k] = tempi[k];
            Gi[k] = -tempr[k];
```

```
161     }
        for ( k=(int)ceil(N/2)+1; k<=N; k++ )
163     {
            Gr[k] = -tempi[k];
165         Gi[k] = tempr[k];
        }
167     if(debug){
            if((fp1=fopen("ffrpt.bin","wb"))!=NULL)    {
169             num1=fwrite(Gr,sizeof(double),row,fp1);
                fclose(fp1);
171         }
            if((fp1=fopen("ffipt.bin","wb"))!=NULL)    {
173             num1=fwrite(Gi,sizeof(double),row,fp1);
                fclose(fp1);
175         }
        }
177     free (tempr);
        free (tempi);
179 }/*End of PhaseShift() function*/
    void Ifft(void)
181 {
        double *Gt_tmp; /* It takes the real part of R_ifft*/
183     double *Gt_tmpi;
        FILE *fp1;
185     int k,i,z1,num1;
        Gt_tmp = (double *)calloc(sizeof(double),row);
187     Gt_tmpi = (double *)calloc(sizeof(double),row);
        printf("\nPerforming IFFT ... ... ...\n");
189     for (k=0;k<=N;k++){
            Gt_ifft[k].r=Gr[k];
191         Gt_ifft[k].i=Gi[k];
        }
193     z1=ziffts(fftdebug,Gt_ifft,row);/*IFFT of the signal in spatial
            coordinate*/
        printf("\nEnd of IFFT ... ...\n");
195     for (k=0;k<=N;k++) {
            Gt_tmp[k]=Gt_ifft[k].r;
197     }
        if(debug){
199     fp1=fopen("ifft.txt","w");
    if (!fp1)
201 fprintf(stdout,"Cann't write in %s\n",fp1);
    for(i=0; i<=N; i++)
203 fprintf(fp1,"%.4e\n",(Gt_ifft[i].r));
    fclose(fp1);
205 }
        printf("\nPerforming IFFT writing... ... ...\n");
207     if(debug){
            if((fp1=fopen("iffrpt.bin","wb"))!=NULL) {
209             num1=fwrite(Gt_tmp,sizeof(double),row,fp1);
                fclose(fp1);
```

```
211         }
            if((fp1=fopen("iffipt.bin","wb"))!=NULL) {
213             num1=fwrite(Gt_tmpi,sizeof(double),row,fp1);
                fclose(fp1);
215         }
        }
217  free(Gt_tmp );
     free(Gt_tmpi );
219 }/* End Of Function Ifft*/
    int EnvelopeReconstruction(void)
221 {
        FILE *fp1;
223     int k;
        doublecomplex *G; /*Input signal readed from input file in complex
             form*/
225     doublecomplex *Ganalytical;/*Analytical function of our input
            signal*/
        double *test;
227     double *sqrtr;
        double *sqrti;
229     G= (doublecomplex *)calloc(sizeof(doublecomplex),row);
        Ganalytical = (doublecomplex *)calloc(sizeof(doublecomplex),row);
231     test = (double *)calloc(sizeof(double),row);
        sqrtr=(double *)calloc(sizeof(double),row);
233     sqrti=(double *)calloc(sizeof(double),row);
        printf("\nPerforming envelope extraction ... ... ...\n");
235     for (k=0;k<=N;k++){
            RE(G[k]) = Gvalue[k];
237         IM(G[k]) = 0.0;
        }
239     for (k=0;k<=N;k++){
            RE(Ganalytical[k])=G[k].r;
241         IM(Ganalytical[k])=Gt_ifft[k].r;
        }
243     for (k=0;k<=N;k++){
        sqrtr[k]=sqrt(Ganalytical[k].r*Ganalytical[k].r+Ganalytical[k].i*
            Ganalytical[k].i);
245     }
             fp1=fopen("output.txt","w");
247          if (!fp1)
             {
249               fprintf(stdout,"Cann't write extracted envelope in
                      output.txt.\n");
                  free(G);
251               free(Ganalytical);
                  free(test);
253               free(sqrtr);
                  free(sqrti);
255               return 1;
             }
257          for(k=0; k<SampledPoints; k++)
```

```
                fprintf(fp1,"%e,%e\n",Gtime[k],sqrtr[k]);
259         printf("\nExtracted envelope is written in %s\n","output.txt
                ");
            fclose(fp1);
261     free(G);
        free(Ganalytical);
263     free(test);
        free(sqrtr);
265     free(sqrti);
        return 0;
267 }
    /*Main Function*/
269 int main_hilbert(int argc, char *argv[])
    {
271     int status=0,i=1;
        char fname[256],c;
273     if(argc==2)
        {
275         printf("\nInput File Name: ");
            /* scanf("%s,InputFileName"); */
277         strcpy(fname, argv[1]);
            InputFileName= fname;
279         printf("%s\n",InputFileName);
        }
281     else
        {
283     printf("\nUse default file : %s\n", InputFileName);
        }
285     /* Reading the sampled data*/
        do
287     {
            N=(int)pow(2,i)-1;
289         i++;
        }while (MAX_POINT > N);
291     if (debug)
        {
293     printf("N= %d\n",N);
        }
295     row=N+1;
        Gvalue = (double *)calloc(sizeof(double),row);
297     Gtime = (double *)calloc(sizeof(double),row);
        Gr = (double *)calloc(sizeof(double),row);
299     Gi = (double *)calloc(sizeof(double),row);
        Gt_ifft= (doublecomplex *)calloc(sizeof(doublecomplex),row);
301     Gc = (double *)calloc(sizeof(double),row);
        status = ReadData();
303     if (status== 1) goto MainExit;
        /*Does FFT*/
305  Fft();
        /*Appropriate Phahe has been Shifted*/
307  PhaseShifting();
```

```
    /*Does IFFT*/
309 Ifft();
    /*Envelope Reconstruction */
311 status = EnvelopeReconstruction();
    if (status== 1) goto MainExit;
313 printf("\n\n==========\n\n");
    printf("Input file name : %s \n",InputFileName);
315 printf("Output file name output.txt\n");
    printf("\n==========\n\n");
317 MainExit:
    free(Gvalue);
319 free(Gtime);
    free(Gr);
321 free(Gi);
    free(Gt_ifft);
323 free(Gc);
    printf("\n\nPress any key to exit.\n");
325 scanf("%c",&c);
    return (0);
327 }/*End Of Main*/
```

List of Figures

1.1 Envelope of a modulated cosine wave 6
1.2 Impulse response of a discrete-time Hilbert transformer 11

2.1 Frequency response of a rectangle window 15
2.2 Transfer function of a discrete-time Hilbert transformer 16
2.3 Block diagram of the algorithm under evaluation as described in [ISO] 22
2.4 Flowchart of the algorithm under evaluation as described in [ISO] . 23
2.5 Block diagram of the error measurement of the code in [ISO]. It shows L branches from L realisations where the errors of the C-code against a Matlab® version are calculated. From the results we get the sample mean $\hat{\boldsymbol{\mu}}_{\boldsymbol{\Delta Y}}$ which is still a vector. The histogram of this sample mean is plotted in Figure 2.6. 26
2.6 Histogram of the resulting noise of the implemented algorithm . . . 27
2.7 Block diagram of the error measurement of a FFT and IFFT routine used in this paper. 28
2.8 FFT: Estimated power spectrum density of the error 29
2.9 FFT of the impulse response of a Hilbert transformer ("rectangular window") 33
2.10 FFT of an Optimal FIR filter of a Hilbert transformer 33
2.11 Block diagram of simulation. 34
2.12 Baseband transmit signal 35
2.13 Ideal and detected envelopes of the second symbol (Sequence X) . . 36
2.14 First jump between the amplitudes of 0 and 1 of ideal envelope and real-world envelope. The graphic of the envelope $r(t)$ is here the same for a rectangular window, Optimal FIR filter or Blackman window since the differences are too small for the plot! t_r specifies the rise time, and h_{ovs} the overshoot (cf. with Appendix C). 36
2.15 First fall between the amplitudes of 1 and 0 of ideal envelope and real-world envelope. The graphic of the envelope $r(t)$ is here the same for a rectangular window, Optimal FIR filter or Blackman window since the differences are too small for the plot! t_f specifies the fall time and h_{ovs} the overshoot (cf. with Appendix C). 37
2.16 Difference between rectangular window, Blackman window and Optimal FIR filter 37

2.17 Difference (Error) between calculation of the envelope with rectangular window and ideal envelope. 38
2.18 Difference (Error) between calculation of the envelope with Optimal filter and ideal envelope. Cf. with Figure 2.17 to see the improvement from the Optimal FIR filter. 39
2.19 Spectrum of the analytic signal 40
2.20 Errors between detected envelope using a Blackman window and ideal FIR filter 41

4.1 Blockdiagram of the source code model 50
4.2 Blockdiagram of the measurement chain model 50

5.1 Blockdiagram of the testbed 58
5.2 Input-output-relation with different values of IIP3. The star (*) denotes that the signal is normalized. The typical interval of the input signal without noise and errors is $[-1; 1]$ in our simulations. 62
5.3 Input-power versus output-power simulated with an IIP3 of 30 dBm*. 63
5.4 Coarse flowchart of the developed simulations. 66

6.1 Spectrum of signal 68
6.2 Time-signal without 2^{nd} and 3^{th} harmonics 69
6.3 Time-signal with 2^{nd} and 3^{th} harmonics 70
6.4 Time-signal with harmonics, zoomed 70

7.1 Error of different second and third harmonics 71
7.2 The error vs. IIP3. 72
7.3 Error introduced by different sampling rates. 73
7.4 Error introduced by Gaussian noise. 74
7.5 Quantization error for different Gaussian noise. 74
7.6 Error introduced by time jitter. 75
7.7 Different phase shifts of second and third harmonics. 76
7.8 Influence of different bandwidths of Gaussian lowpass filter without second and third harmonics and without noise. 77
7.9 Influence of different bandwidths of Gaussian lowpass filter without second and third harmonics and with noise. 78
7.10 Influence of different bandwidths of Gaussian lowpass filter with second and third harmonics and without noise. 78
7.11 Influence of different bandwidths of Gaussian lowpass filter with second and third harmonics and with noise. 79
7.12 Influence of different bandwidths of Gaussian lowpass filter with second and third harmonics, noise and four different sample frequencies. Note that the sample phase is random. 79

7.13 Influence of different bandwidths of Gaussian lowpass filter with noise and four different sample frequencies. Note that the sample phase is random. 80

.

List of Tables

2.1 Parameters for the generalized cosine window 16
2.2 Confidence interval of the error of the code under evaluation 27
2.3 Difference of `float` and `double` data type 30

3.1 Calculation of the bandwidth $f_{-3\,\mathrm{dB}}$ of a signal. 48
3.2 Calculated values of the bandwidth $f_{-3\,\mathrm{dB}}$ of the signal defined in [ISO]. 48

6.1 Parameters of the source model used in these simulations 67
6.2 Setup of the simulations of Agilent's DSO7052A [Agi09]. 69

Bibliography

[agia] Choosing an oscilloscope with the right bandwidth for your application. Technical report, Agilent Technologies.

[agib] Evaluating oscilloscope sample rates vs. sampling fidelity: How to make the most accurate digital measurements. Technical report, Agilent Technologies.

[Agi09] Agilent Technologies, Inc. *Agilent Technologies - InfiniiVision 7000 Series Oscilloscopes*, January 2009.

[Ber07] Greg Berchin. Precise filter design. *IEEE Signal Processing Magazine*, pages 137–139, 2007.

[Che06] Jesse E. Chen. Modeling rf systems. *The Designer Guide Community*, May 2006.

[dob08] *"Zeitdiskrete Signale und Systeme, eine Einführung in die grundlegenden Methoden der digitalen Signalverarbeitung"*. J. Schlembach Fachverlag, 2008.

[Dug58] J. Dugundji. "envelopes and pre-envelopes of real waveforms". *"Information Theory, IRE Transactions on"*, 4(1):53–57, March 1958.

[FJ05] Matteo Frigo and Steven G. Johnson. "ieee, vol. 93, no. 2, pp. 216-231: The design and implementation of fftw3". Technical report, "IEEE", 2005.

[fli08] *Signale und Systeme - Grundlagen und Anwendungen mit Matlab*. J. Schlembach Fachverlag, 2008.

[flo85] "ieee standard for binary floating-point arithmetic". *"ANSI/IEEE Std 754-1985"*, pages –, Aug 1985.

[GN91] R.M. Gray and D.L. Neuhoff. Quantization. *"ECSC-2, Proceedings of the Second European Conference on Satellite Communications"*, Oct 1991.

[hel07] Mathworks' MATLAB® help, 2007. http://www.mathworks.com/.

[hla08] *Class notes of Stochastic Signals*. 2008.

[ISO] "iso/iec 10373-6:2001, amendment 7: Test methods for epassport ". Technical report, "ISO/IEC".

[ISO08] Iso/iec cd 14443-2.fcd: Identification cards — contactless integrated circuit(s) cards — proximity cards — part 2: Radio frquency power and signal interface. Technical report, ISO/IEC, 2008.

[KH59] J. Klapper and C. Harris. On the response and approximation of gaussian filters. *"Audio, IRE Transactions on"*, 7(3):80–87, May 1959.

[KL71] Toyohisa Kaneko and Bede Liu. On local roundoff errors in floating-point arithmetic. Technical report, 1971.

[mit02] *Digital signal processing - a computer-based approach*. Mc. Graw Hill, 2002.

[opp92] *Zeitdiskrete Signalverarbeitung*. Oldenbourg, 1992.

[pro96] *Digital signal proecessing - principles, algorithms, and applications*. Prentice-Hall International, Inc., 1996.

[Ram71] Georg U. Ramos. "math. comput., vol. 25, pp. 757-768: Roundoff error analysis of the fast fourier transform". Technical report, Stanford, CA, USA, 1971.

[Rap98] Christian Rapp. "effects of hpa-nonlinearity on a 4-dpsk/ofdm-signal for a digital sound broadcasting system.". *"Information Theory, IEEE Transactions on"*, 44(6):2325–2383, Oct 1998.

[Rap02] *"Wireless Communications Principles and Practice, second edition"*. Prentice Hall, 2002.

[SMMW00] F. Sthal, M. Mourey, F. Marionnet, and W.F. Walls. "phase noise measurements of 10-mhz bva quartz crystal resonators". *"Ultrasonics, Ferroelectrics and Frequency Control, IEEE Transactions on"*, 47(2):369–373, Mar 2000.

[ste96] *Signal processing algorithms in Matlab*. "Prentice Hall P T R Prentice-Hall, Inc. ", 1996.

[TL77] Tran Thong and B. Liu. "accumulation of roundoff errors in floating point fft". *Circuits and Systems, IEEE Transactions on*, 24(3):132–143, Mar 1977.

Index

aliasing, 45, 55
Analytic Signal
  definition, 7
  Spectrum, 7, 39

complex envelope, 7
confidence interval, 82
convolution, 7
correlation, 18, 19

data types
  double, 28
    errors of an addition, 30
    errors of an multiplication, 29
  float, 28

envelope, 5

Fast Fourier Transform
  definition, 18
  errors, 14, 27, 29
filter design
  Frequency-Domain Least-Squares, 17
  Optimal Filters, 16
Fourier transform
  continuous, 7
  discrete, 18
  discrete-time, 10
  fast Fourier transform, 18
frequency response
  flat, 47
  Gaussian, 47
Frequency-Domain Least-Squares Filter Design, 17

Gibbs phenomenon, 13, 31
  Blackman window, 16, 40
  Discontinuities, 13
  error floor, 71
  Frequency-Domain Least-Squares, 17
  Generalized cosine window, 15
  Leakage, 14
  Optimal Filters, 16
  Optimal filters, 31
  Window sequence, 14

input third order intercept point, 54, 61, 72

jump function
  overshoot, 84
  rise time, 84
  settle time, 84

noise, 53
  AWGN, 53
  single sideband noise spectral density, 51, 59
non-linearity system, 52

Optimal Filters, 16
Optimal filters, 31

PCD, 53
pre-envelope, 7
pulse amplitude modulation, 83

quantization, 56, 72, 73
quantization interval, 56

sampling, 45
  aperture jitter, 49

aperture time, 48
equivalent-time, 46
in simulation, 55
real-time, 46
sequential, 46
single shot, 46
signal-to-noise ratio, 73
spectrum
energy spectrum density, 18
power spectrum density, 20
stochastic, 18
estimation of moments, 20
moments, 19
statistic distributions, 21

Taylor series, 52
third order intercept point, 54

Zero-padding, 23

Nomenclature

$:=$	"defined as"
$[\,\cdot\,]$	(physical) unit of (physical) quantity or nearest integer function.
$.^T$	"transposed of"
$\equiv$	"identity of"
$\mathrm{fl}\{\,\cdot\,\}$	"floating-point result of"
$\forall$	"for all"
$\mathcal{F}\{\,\cdot\,\}$	Fourier transform, discrete time Fourier transform, FFT or DFT
$\hat{\cdot}$	estimated quantity
$\mathrm{Im}(\,\cdot\,)$	"imaginary part of"
$\mathbb{I}_k$	$\{0 \cdots 2^k - 1\}$ with a const. k
$\mathbb{N}$	set of natural numbers
$\mathbb{Q}$	set of rational numbers
$\mathbb{Q}_{64}$	subset of rational numbers defined by 64-Bit data type `double`. $\mathbb{Q}_{64} \subset \mathbb{Q}$
$\mathbb{R}$	set of real numbers
$\mathbb{Z}$	set of integers
$\mathcal{A}\{\,\cdot\,\}$	"Analytic Signal of"
$\mathcal{H}\{\,\cdot\,\}$	"Hilbert transform of"
$\boldsymbol{X}$	matrix
μ_X	mean of random variable X
$\mathrm{Re}(\,\cdot\,)$	"real part of"
$\rightarrow$	"tends to"

σ standard deviation

$\sigma(j\omega)$ Heaviside step

σ_X^2 variance of of random variable X

$\operatorname{sign}(\,\cdot\,)$

$\star$ convolution

dBc $\frac{P}{1\text{ dBc}} := 10 \cdot \log\left(\frac{P}{P_{\text{carrier}}}\right)$

dBm$*$ $\frac{P}{1\text{ dBm}*} := 10 \cdot \log\left(1000P\right)$ with $[P] = 1$ (normalized signal)

$\boldsymbol{x}$ vector

$\{\,\cdot\,\}$ sequence or set

$f^{-1}(\,\cdot\,)$ inverse of $f(\,\cdot\,)$

$F_X(x)$ Cumulative Distribution Function of X

$f_X(x)$ Probability Density Function of X

$m_{X_1,X_2}^{(k_1,k_2)}$ mixed moment of random variables X_1 and X_2 with order k_1 and k_2

$m_X^{(k)}$ k^{th} moment of random variable X

$P\{\,\cdot\,\}$ "Propability of"

R reference resistance or resample factor

$R[n_1, n_2] \equiv R_{X[n_1],X[n_2]}$ autocorrelation

$r_X[m]$ wide-sense stationarity autocorrelation

$S_X(e^{j\Theta})$ power spectrum density

$X \sim \mathcal{N}(\mu_X, \sigma_X^2)$ normal/Gaussian distribution of random variable X

$X(e^{j\Theta})$ "discrete-time Fourier transform of x[n]"

$X(t)$ random variable

$x(t)$ continuous-time signal

$X[n]$ "discrete Fourier transform of x[n]"

$x[n]$ descrete-time signal with time index n

AWGN Additive White Noise

cdf Cumulative Distribution Function

CI confidence interval

DFT Discrete Fourier Transformation

FDLS frequency-domain least-squares

FFT Fast Fourier Transform

FIR Finite Impulse Response

HT Hilbert transform

IDFT Inverse Discrete Fourier Transformation

IFFT Inverse Fast Fourier Transform

IIP3 third order input intercept point

IP3 third order intercept point

LTI Linear Time Invariante

PAM pulse amplitude modulation

PCD Proximity Coupling Device

pdf Probability Density Function

psd Power Spectrum Density

RFID Radio Frequency Identification

SNR signal-to-noise ratio